Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2014, Article ID 124872, 8 pages
http://dx.doi.org/10.1155/2014/124872

Research Article

Merging Agents and Cloud Services in Industrial Applications

Francisco P. Maturana,[1] Juan L. Asenjo,[2] Neethu S. Philip,[3] and Shweta Chatrola[4]

[1] *Control Architecture and Technology, Rockwell Automation, Cleveland, OH, USA*
[2] *Customer Support and Maintenance, Rockwell Automation, Cleveland, OH, USA*
[3] *Washkewicz College of Engineering, Cleveland State University, Cleveland, OH, USA*
[4] *Monte Ahuja College of Business, Cleveland State University, Cleveland, OH, USA*

Correspondence should be addressed to Francisco P. Maturana; fpmaturana@ra.rockwell.com

Received 4 December 2013; Accepted 10 July 2014; Published 19 August 2014

Academic Editor: Yongqing Yang

A novel idea to combine agent technology and cloud computing for monitoring a plant floor system is presented. Cloud infrastructure has been leveraged as the main mechanism for hosting the data and processing needs of a modern industrial information system. The cloud offers unlimited storage and data processing in a near real-time fashion. This paper presents a software-as-a-service (SaaS) architecture for augmenting industrial plant-floor reporting capabilities. This reporting capability has been architected using networked agents, worker roles, and scripts for building a scalable data pipeline and analytics system.

1. Introduction

Industrial automation has come a long way from the early days of electromechanical systems to the distributed, modular, and intelligent control of applications that we have today where the automation controllers feature multicore processors, rich set of capabilities, and so forth, pushing the envelope. Improvements on the hardware and features are not enough as the increasing complexity of applications and information require a different paradigm, one that promotes the collaboration of distributed autonomous agent platforms. It is believed that agents will provide the foundation for a collaborative approach to a supervisory control layer with autonomous capabilities and access to vast amounts of information with higher-order responsible in the decision making layers of the automation enterprise [1, 2]. Agents are triggered when an event that requires complex computation occurs in the physical world. Agents have to do with business intelligence rules and a mix of device- and system-level information. Now that data can be stored in the so called big data storage, how to merge the agent capabilities with the cloud in a harmonious information processing system is one of the challenges of this work.

A lot of research has been conducted in agent-based cloud computing applications [3, 4]. A self-organized agent-based service composition framework is discussed which uses agent-based problem solving techniques such as acquaintance networks and contract net protocol [5]. This method can be used in scenarios involving incomplete information about cloud participants. The solution described in this paper also leverages the contract net protocol to solve multiagent event handling tasks.

In [6] a multiagent model for social media service based on intelligence virtualization rules is discussed. Intelligence multiagent for resource virtualization (IMAV) manages cloud computing resources in real-time and adjusts the resources according to user behavior.

A new cloud computing architecture for discovery and selection of web services with higher precision is presented in [7]. It is based on the concept of OCCF (open cloud computing federation) which incorporates several CCSPs (cloud computing service providers) to provide a uniform resource interface for the clients. This offers the advantages of unlimited scalability, availability of resources, democratization of cloud computing market, and reduced cost to the clients.

The solution described in this paper leverages the above concepts to uniquely solve industrial automation domain specific problems. Security of data is a prime concern for all users alike. Cloud computing infrastructure ensures data

security and sovereignty through service level agreements (SLA's) agreed mutually by the contending parties. The infrastructure described here leverages this capability of the cloud computing technology and thus offers a secure environment for each customer to operate in parallel. The application illustrated relies on Internet connectivity between the agents, analytics orchestrator, and cloud services. However, the system also accommodates network failures, using store and forward mechanisms by which the messages are stored during a period of network failure and forwarded once the connection is regained.

Given the nature of a distributed agent framework and the advent of cloud computing there is an opportunity to merge these capabilities into a highly scalable distributed architecture where stream processing and MapReduce paradigms provides new possibilities for distributed agent architectures. Our proposition is to layer an agent framework on top of multitenant cloud systems where high volume of data is ingested in real-time, large sets of historical data (petabytes+) become available for stream processing, and these powerful capabilities can be leveraged in a distributed agent framework.

The data collected from machines and applications are growing at very large rates leading to torrential volumes of data. As data grows, the physical infrastructure and computing capabilities were required to maintain the data scales up, incurring into increased cost to maintain the data. But the focus needs to be shifted from maintaining data to utilizing the data for enhancing business and operations intelligence and to solve domain specific problems. Thus adopting cloud computing technology in the industrial automation domain is a very viable option for lowering the overall cost while gaining increased computing capacity and big data storage.

Here a framework is described that interfaces intelligent agents and cloud analytics services to perform complex analytics on the data collected from on premise applications.

2. Data Pipeline Architecture

The cloud is all about "unlimited" storage and compute resources to process "big data" (volume, velocity, and variety). Our cloud infrastructure essentially provides a data pipeline between on premise devices and cloud applications. This pipeline has the ability to negotiate storage with the cloud level application to scale up storage capacity as needed by the emitting system (elastic cloud).

To take advantage of Microsoft Azure PaaS (Platform as a Service), the main focus was to achieve a high volume, high velocity, and high variety of data transferring in a highly secure fashion. Once the data is made available to Azure, it is consumed using native Microsoft and open source technology stack for dashboards, batch/stream processing, and analytics (including Hadoop ecosystem, Cloud ML, Power BI, Excel, and Sql reporting system).

2.1. Azure Storage Fabric. Microsoft Azure Storage Fabric is being leveraged as the main data management channel to provide unlimited storage capacity that will scale up automatically and natively (currently measured as providing 1,000,000 of hits per second for concurrent users pushing data). Azure Chunking technology has also been leveraged to partition a large number of records into smaller and manageable chunks (1,000,000 records typically). There is a data type concept being leveraged into the structuring of the data at the data collection point on premise. This data type has evolved into a framework that allows concurrent data collectors with shared resources like serialization, compression, and queuing. At present, there are alarms, live data, historical data, Sql datasets, and CIP messaging collector types implemented.

PowerShell scripting is another native Microsoft component that helps to maximize data ingestion flexibility and transferring velocity. Microsoft Azure technology has several merits that have been proven very relevant and effective into industry-based data collection and storage system.

To give unique identity to each data collector agent, Azure Fabric Security was augmented with a Troll Bridge service in addition to HTTPS and certificates (from a certificate authority). The Troll Bridge service defines access policies and completely removes access keys from the on premise locales. In the cloud, cloud users and applications are supported by native trust centers by providing a unique identity to each data collector agent.

The cloud pipeline has been designed in such a way that the storage for the incoming data is Manifest driven. The Manifest identifies storage destinations like Sql DB (which permits up to 500 GB of continues storage), Sql Server on premise or in the cloud as IaaS (Infrastructure as a Service), or blob storage for HDInsight (Hadoop). The same Manifest also maps data types to analytics. For example, if the data is mapped to Hadoop (HDInsight on Azure), the data can then be processed using the Hadoop ecosystem for analytics which includes high-level languages such as Pig, Hive, MapReduce, Language R, and HBase. Moreover, the cloud pipeline scales out natively and so far no limitations have been observed during the processing of the data types, storage volume, connectivity to dashboard systems, or analytics.

2.2. Worker Role. The worker role is basically an analytics script that has been designed to interpret an application-specific manifest and to load application-specific functionality into the processing of the data based on the manifest directions, as shown in Figure 1.

The worker role processes the event notifications/requests from the on premise cloud agent according to a given priority queue set. The manifest to be interpreted is identified in the event header. The manifest conveys the tenant's information which produced the data. It also indicates the analytics assembly module to be loaded to process the data. The manifest contains a reference to other manifests for tags and metrics descriptions. Tag manifest contains a list of tags on which the analytics is to be performed. Metrics manifest specifies coefficients, threshold, and constants that are required for executing the tenant specific calculations.

When the worker role receives a notification about new data, it fetches an assembly object to convert the data in the transient blob from raw unstructured form to a structured form prior to analytics processing. This conversion transfers

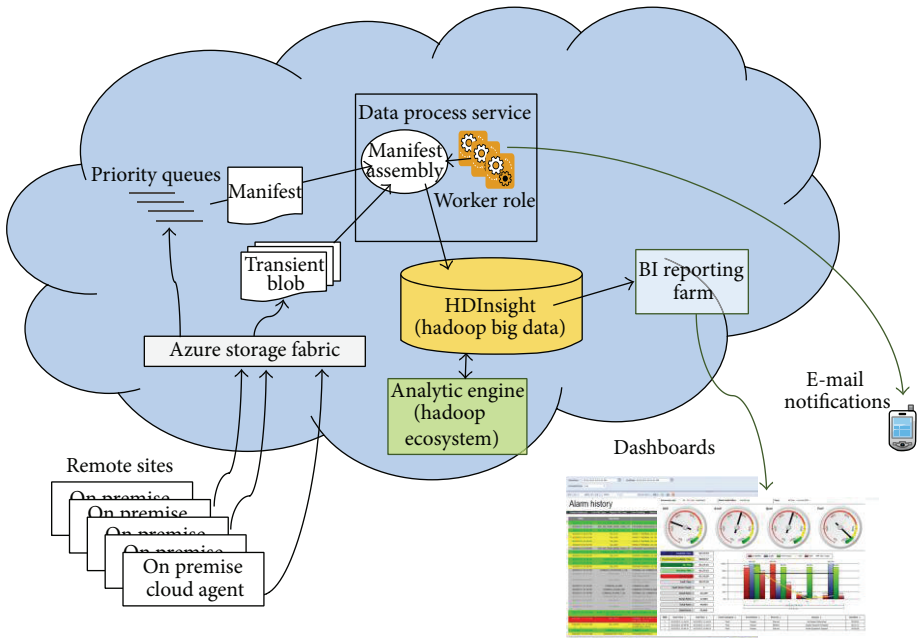

Figure 1: Cloud-based Solution.

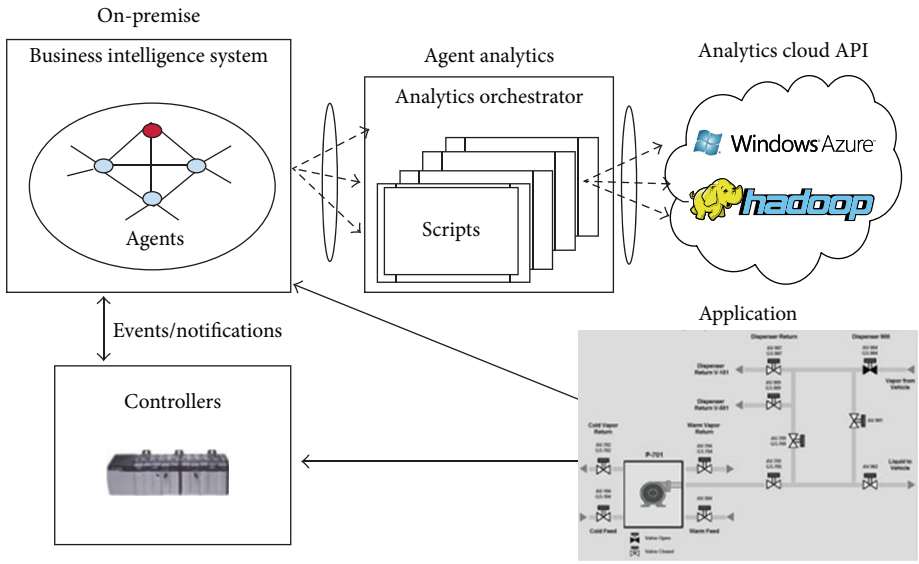

Figure 2: Data ingestion framework.

the data into Sql space and it then is moved to a manifest-defined database. The system consists of multiple assemblies that are designed to perform various analytics on the stored data. Here the tenant can decide what analytics needs to be used locally and which can also include interfacing with remote cloud analytics.

3. Data Ingestion Framework

There is a data ingestion framework for encapsulating various functions for connecting plant floor level controllers (the actual source of data) with the cloud level data ingestion mechanism, as shown in Figure 2.

In this framework, there is an application level system composed of physical hardware, sensors, networks, and so forth, all interconnected to perform a process or activity towards producing a product or generating a resource. Controllers contain control logic for processing sensor data and to effect control commands into the hardware. The controllers collect the process data in the form of tags in a tag-table image (database). Agent software encapsulates business intelligence rules and they also possess the ability to reason about those rules with other agents in a cooperative decision making network.

Analytics orchestrator is an intelligent router that routes the requests from the agents to the cloud analytics services.

The communication between agents and analytics orchestrator and analytics orchestrator and cloud are based on contracts. The agents are generally on the premises and they interact with the controllers that control the application. The analytics orchestrator can be on premise or in the cloud. The analytics orchestrator communicates with cloud analytics via a restful interface.

3.1. Agent Module. Agents are intelligent modules that make decisions on behalf of the controllers. The agents consist of a control algorithm, a reasoning part, and data table image interface. The control algorithm is dedicated to monitoring the machine status and giving signals to actuators that control the system in real-time. There are agents associated with controlling components. These two layers communicate with each other via events and tag data read/write.

The agents evaluate the control scenarios that are given by the controllers (a control scenario includes data and events). The agents attach semantic context to the scenario to reasoning around the business rules. In a collaborative form, the agents exchange information to negotiate and converge to common and suitable decisions about their scenarios and ultimately for steering control. Each agent is assigned with a capability and a capability can comprise multiple operations. All capabilities are registered with a directory service which provides social organization and discovery to the system. When a situation occurs in the application that requires high-level reasoning, an event is triggered into the agent level. For example, a controller may need temperature gradient calculation in order to select a control action for a pressure release valve. This request depends on a more complex calculation that involves enthalpy tables and interpolations of sensor data. Sensor data that has been accumulated in big data storage is then invoked to perform the calculations. Neither the agents nor the controllers have all the data they need to do the calculations.

The agents solve the event handling for tasks using the Contract Net Protocol. It is a task sharing protocol in multi-agent systems, consisting of a collection of agents that form a contract net. The agents with the particular capability will analyze the request from the controller and will generate a request to be sent to the analytics orchestrator. The request is generated based on a contract that is already defined between the agents and the analytics orchestrator. Based on the contract the agents will include all the information required by the analytics orchestrator to route the request to the specified analytics provider. The agents wait for response from the analytics orchestrator.

3.2. Analytics Orchestrator. The analytics orchestrator listens for incoming requests from the agents. When a request is received, it verifies that it is a new request or if a process already exists for it. The request is discarded if there is a process with the same id already handling it. If it is a new request, a thread is created to handle it. The request is parsed to extract routing information into the data analytics that is required for the current calculation that is requested by the agents. The analytics orchestrator supports a mix of scripts that can be written in high-level languages such as Pig, JavaScript, Hive,

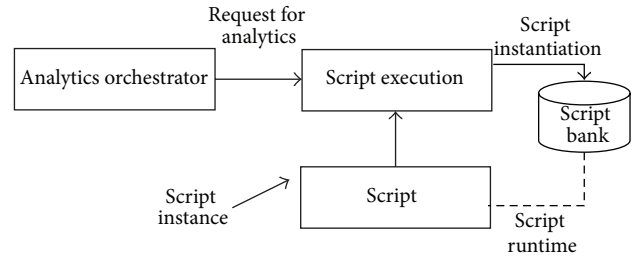

Figure 3: Framework script structure.

and so forth. The scripts are maintained in a script bank, as shown in Figure 3.

Each script inherits from the script execution which manages script class instantiation throughout its class inheritance chain. Once instantiated, the script further parses the original request to extract analytics specific parameters. The user specifies that analytics should be performed with the data and other coefficients and threshold data required to process the request. The caveats in this script structure is that the only modifiable part when adding more application specific functions is at the script level only. This step requires writing a new script and adding it to the bank.

A request is then generated to the requested cloud analytics provider which is specified by user. The request is created based on a contract between the analytics orchestrator and cloud provider. The request is sent to the web service client interface that invokes the requested cloud analytics service.

3.3. Web Service Interface. The cloud analytics scripts are deployed as web services and can be invoked by the client web service interface. The web services are based on REST architecture. In the REST architectural style, data and functionality are considered as resources and are accessed as uniform resource identifiers (URIs). REST is designed to use stateless communication protocol such as HTTP. So the client web service interface connects to the requested web service by invoking the URI (uniform resource identifier) to the web service which is specified by the user. The communication is synchronously blocked which means that the requesting thread is blocked till it receives a response from the cloud analytics web service or waiting time expires.

3.4. Cloud Analytics Provider. Due to the high end computing capabilities required for performing such a large scale of analytics the cloud analytics capabilities are leveraged in this architecture. Cloud providers use technologies such as NoSQL databases, Hadoop, and Map Reduce to perform analytics on Big Data.

The cloud web service supports multitenancy; that is, it allows multiple customers to access multiple instances of single software. There are SLA (service level agreement) that defines the rules and regulations of the communication between the cloud provider and an external user. Thus the cloud provider ensures data security and sovereignty in each customer environment. Here once the cloud provider receives a request, it analyzes the request and invokes the analytics script that will execute analytics and retrieve the requested

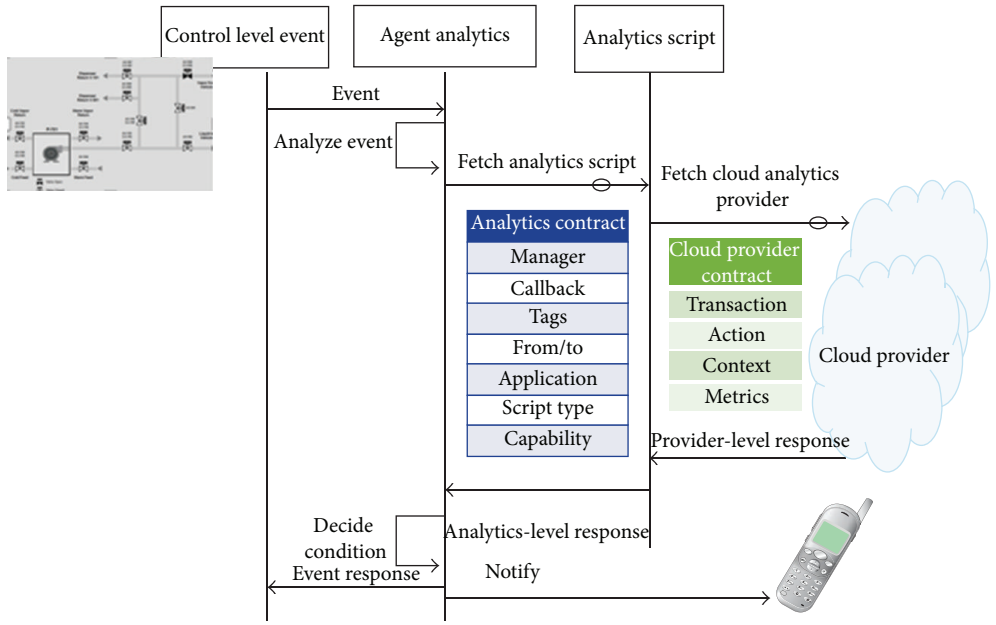

Figure 4: Design and programming phase.

information. The response is then packed in to a predefined response format and sent back to the analytics script that raised the request. If any error conditions have occurred, the error conditions are also specified along with the response.

4. Programming Model

The programming model that is used to develop applications on the defined infrastructure consists of three phases: (1) design and programming, (2) runtime, and (3) offline processing.

4.1. Design and Programming Phase. In this phase, the user designs and develops models and scripts that will be executed during the run time phase. As shown in Figure 2, the application control logic is modeled into the controllers and the business intelligence logic is modeled by the agents. The user also develops domain specific analytics to be executed. For example, the user can develop analytics scripts either to generate reports and trends or to proactively control applications or to develop any smart applications on top of the analytics orchestrator infrastructure. The user also defines the contracts for communication between agents and analytics orchestrator and analytics orchestrator and cloud providers.

4.2. Run Time Phase. The sequence of events during the runtime phase is as described in Figure 4. During run time, the application is running on the premises and generates event notifications to the agents. The agents generate requests to the analytics orchestrator guided by business rules. Analytics orchestrator will execute the specified analytics script that was designed by user and generate requests to the cloud analytics provider. The cloud analytics provider will perform the required analytics and respond back to the analytics orchestrator with the required data. The analytics orchestrator analyzes the response and performs the required analytics as described by user in the design phase. The complete response is then sent back to the agent that triggered the request. The agents will take required decision based on the response and convey the decision to the controller and to user through notifications.

Each script inherits from the script execution which manages script class instantiation throughout its class inheritance chain. Once instantiated, the script further parses the original request to extract analytics specific parameters. The user specifies what analytics should be performed with the data and other coefficients and threshold data, required to process the request. The caveats in this script structure is that the only modifiable part when adding more application specific functions is at the script level only. This step requires writing a new script and adding it to the bank.

4.3. Offline Processing Phase. During runtime, anomalies may occur that were not anticipated previously. In the offline processing phase, the user can analyze these anomalies by extracting parameter characteristics from Big Data for an interval of time until the occurrence of the anomaly. The parameter characteristics can be analyzed using cloud computing technologies like Map Reduce. Once the anomaly root cause is determined, new business intelligence rules can be programmed into the analytics scripts that will proactively handle occurrences of the particular behavior in the next deployment. Thus a self-learning system is designed that can continuously adopt new rules without affecting the basic infrastructure of the system.

4.4. Common API Specification. The interface between agents and analytics orchestrator and analytics orchestrator and the cloud provider is established by means of contracts. The contract specifies the information that should be specified

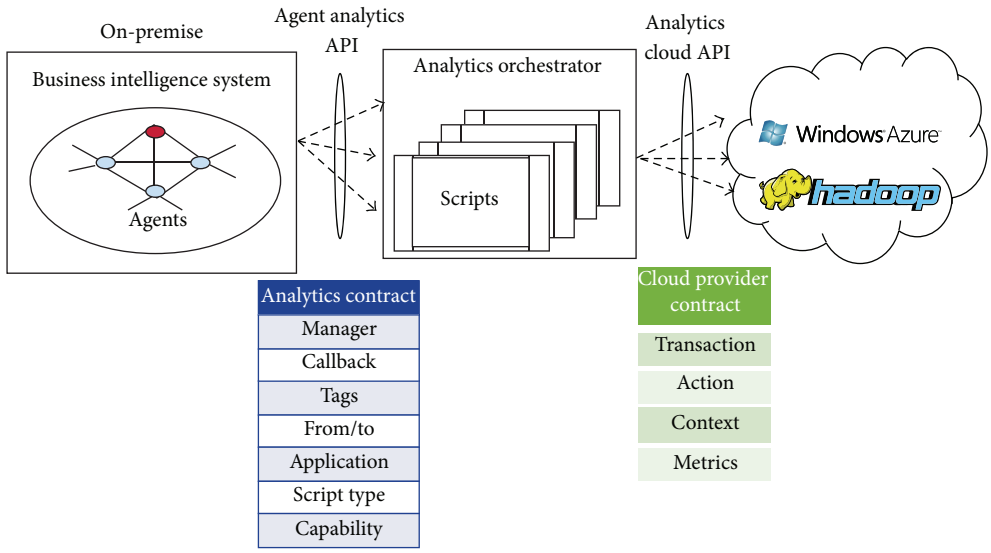

Figure 5: API specification.

by each party when generating a request or response. An example of a contract specification is shown in Figure 5.

The elements in the contract between agents and analytics orchestrator are described below.

(i) Manager identifies the tenant.

(ii) Callback specifies the method to be invoked when response is received.

(iii) Tags specify the parameters on which analytics is to be performed.

(iv) FromTo indicates the time interval for which the data is to be extracted from Big Data storage.

(v) Application specifies the particular application from which the event occurred.

(vi) Script-type specifies the analytics script group to be contacted. There will be one analytics script group per cloud analytics provider. The analytics script group will prepare the requests to the cloud analytics provider according to the specifications of the particular cloud provider.

(vii) Capability represents the analytics to be performed, for example, calculate enthalpy.

The elements in the contract between analytics orchestrator and cloud analytics provider are as follows.

(i) Transaction represents the timestamp of the transaction which is used in analytics to uniquely identify a transaction.

(ii) Action specifies the analytics to be performed by the cloud analytics provider.

(iii) Context specifies the information of the tenant which is used to locate the data of the tenant from the Big Data storage.

(iv) Metrics specify the tag values, time intervals and coefficients, and thresholds information required for executing an analytics.

This is one example of contract that can be defined. The user can modify the details and parameters that need to be included in the contract without affecting the basic infrastructure. The API is defined to be generic enough so that any customer and any application can interact with any cloud provider.

5. Use Case

The selected application is based on a remote monitoring system of a dispenser of natural gas technology. This application generates raw data for 1,000 data points (including temperature, pressure, and flow) every 250 msecs which falls under torrential data.

This application generates up to 120 GB of uncompressed data per month. The goal is to implement the cloud based infrastructure that incorporates data ingestion, storage, and analytics capabilities. The combined effect of such capabilities is intended to provide high-performance data collection and analysis, seamless user interaction, distributed data storage, fault tolerance, and global data availability.

Data ingestion from on-premises to the cloud infrastructure is facilitated by a cloud agent. Figure 6 shows the cloud agent and its relationship to other components at the data collection point. The time series data or tags are collected by software called FTHistorian [8] which acts as an automation-oriented database. FTHistorian provides a rich infrastructure for managing the data as well as local caching. The cloud agent periodically connects to the FTHistorian software to extracts blocks of data from the FTHistorian.

The cloud agent verifies the connectivity between on premise software and the cloud system before sending the data. If a disruption is detected, the cloud agent switches to

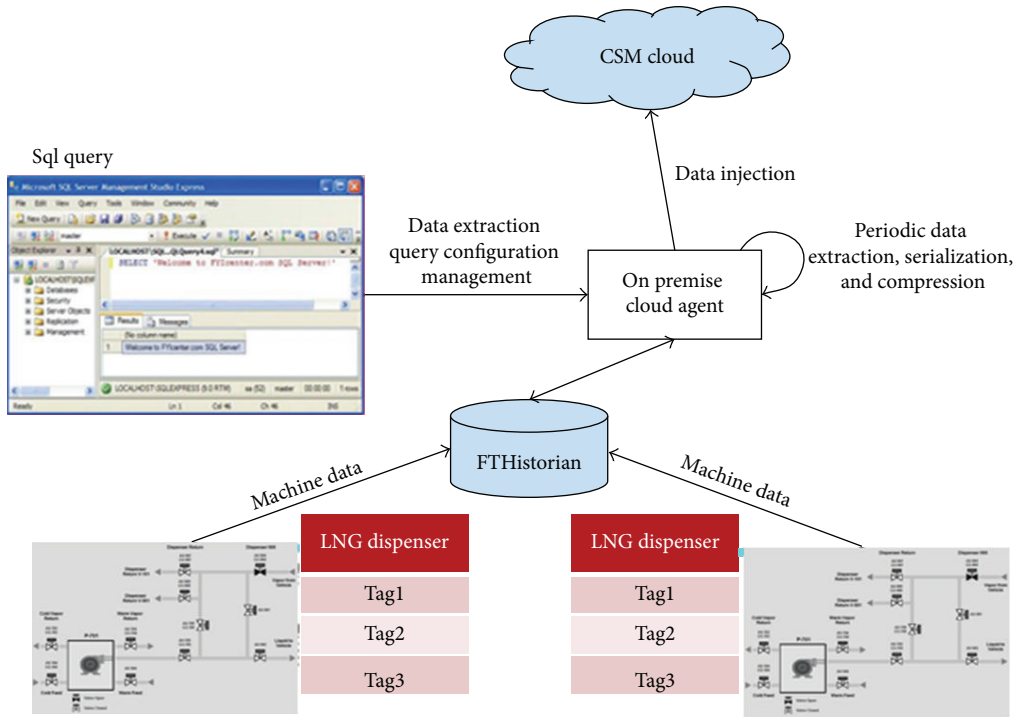

Figure 6: On-premise architecture.

a cache mode until it can reforward the data, after restoring the needed connections and security. When the data reaches the cloud it is stored into transient blob storage and an acknowledgement is sent back to the cloud agent to notify the arrival of the data. The cloud agent then sends a notification to a cloud priority queue system to alert an expecting worker role about the new data that is waiting for processing.

The result of the analytics is displayed to the user through the dashboard, as shown in Figure 7. The figure shows the trends of parameters over a period of time defined by the user. This remote monitoring service can enable business decision makers to thoroughly observe energy conditions and anomalies thus providing them with information to design smarter and energy efficient solutions.

6. Benefits and Assessment

The solution described here leverages the technologies and expertise available to offer services to solve domain specific problems. This section discusses the benefits of implementing this solution.

(a) By utilizing the infrastructure services offered by cloud computing technology, the costs to store and maintain data on premises can be lowered to a great extent. Cloud computing offers on-demand delivery of IT services on a pay-as-you-go pricing. This reduces the time and money required to invest on data centers and other IT infrastructures on premises.

(b) Cloud computing can easily provision resources as and when required. Any number of servers can be procured, be delivered, and be running within no time, thus making it easy to scale up when required.

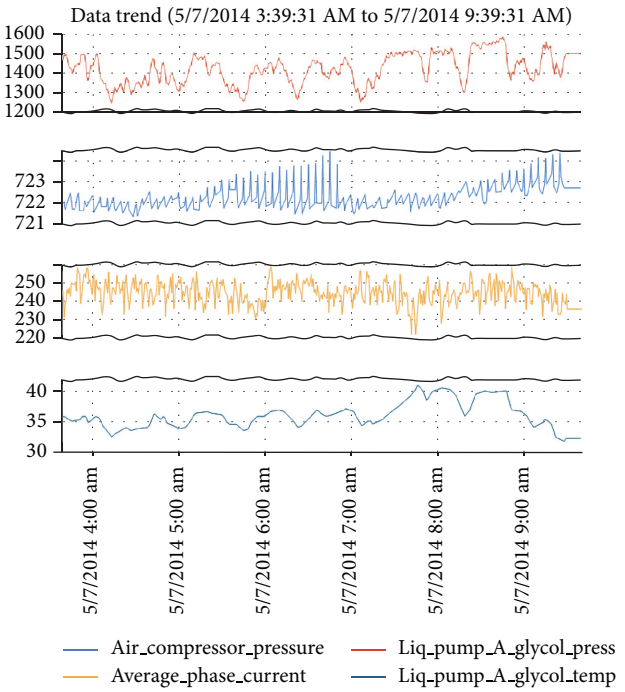

Figure 7: Runtime events data trends.

(c) The industrial automation domain can leverage this cloud computing capabilities to manage the huge amounts of data generated from the sensors and machines. Users can remotely monitor the data from various locations whenever and wherever they want to. The data can be used to develop smart applications

that can proactively control the system or to generate business and operations intelligence.

(d) The infrastructure described here meets the essential requirements of flexibility, scalability, and reusability and can thus be adopted and interfaced with existing systems with ease. Thus the merging of the industrial control technology with cloud computing technology comes with the benefits of lower costs, higher computing capacities, and huge storage facilities which are the need of the hour in the industrial automation domain.

7. Conclusions

Industrial domain applications are increasing in complexity and size, thus calling for the need to scale out analytics capabilities of industrial control services. The data being collected from machines at the premises is becoming more cumbersome and expensive to maintain. Thus services offered by cloud computing technology to store and maintain data are leveraged to meet the ever increasing demand for storage and computing capabilities.

An infrastructure that merges agents with cloud computing technology has been presented as a viable methodology to cope with large volume information in near real-time terms. This infrastructure offers a smart layer supported by agents that can be made in full alignment with automation and control systems. An analytics orchestrator uses a scalable, generic, and flexible API so that any domain user can easily interface to the orchestrator to program analytics and interface with other cloud services.

A real life application was used to validate the remote monitoring system. The solution presented here taps into a strategic partnership between industrial control expertise and cloud computing expertise and the domain-user expertise to solve critical aspects of existing industrial automation platforms.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] F. P. Maturana, R. J. Staron, D. L. Carnahan, and K. A. Loparo, "Distributed control concepts for future power grids," in *Proceedings of the IEEE Energytech*, pp. 1–6, Cleveland, Ohio, USA, 2013.

[2] F. P. Maturana, R. J. Staron, D. L. Carnahan, and K. A. Loparo, "Agent-based test bed simulator for powergrid modeling and control," *IEEE Energytech*, 2012.

[3] W. Xu, B. Wang, and J. Huang, "Cloud computing and its key techniques," in *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering (CSAE '11)*, pp. 404–410, Shanghai, China, June 2011.

[4] P. Patel, A. Ranabahu, and A. Sheth, "Service level agreement in cloud computing," http://corescholar.libraries.wright.edu/.

[5] S. Hamza, K. Okba, and B. Aicha-Nabila, "A new cloud computing framework based on mobile agents for web services discovery and selection," in *Proceedings of the 13th International Arab Conference on Information Technology*, 2012.

[6] M. Kim, H. Lee, H. Yoon, J. I. Kim, and H. S. Kim, *IMAV: An Intelligent Multi-Agent Model Based on Cloud Computing for Resource Virtualization*, International Association of Computer Science and Information Technology (IACSIT), 2013.

[7] J. O. Gutierrez-Garcia and K.-M. Sim, "Self-organizing agents for service composition in cloud computing," in *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom '10)*, pp. 59–66, Indianapolis, Ind, USA, November-December 2010.

[8] "Data Management: Rockwell Factory Historian," Rockwell Automation, 2013, http://www.rockwellautomation.com/rockwellsoftware/data/historian/overview.page.