Edge Computing Architecture for applying AI to IoT

Seraphin B. Calo, Maroun Touna, Dinesh C. Verma
IBM T. J. Watson Research Center
Yorktown Heights, NY, USA
{scalo, touma, dverma}@us.ibm.com

Alan Cullen
BAE Systems
Chelmsford, UK
alan.m.cullen@baesystems.com

***Abstract*— The proliferation of connected IoT devices creates a big data problem for AI based approaches. The response time required by such devices necessitates IoT data to be processed at the edge, but the edge typically lacks the resources to learn the AI models. We present an architecture which preserves the advantages of both edge processing and server-based/cloud-centric computing for AI algorithms. We discuss how policy management can be used to improve and support this architecture.**

Keywords—IoT, AI, streaming data, edge computing, cognitive analytics

I. Introduction

The Internet of Things (IoT) has become important because it can provide ubiquitous services based on real time contextual information. IoT devices generate a significant amount of data due to the large number of devices and the growing use of bandwidth intensive modalities like video. In order to extract insights from high volume IoT data in real time, processing needs to happen near the location where the data is generated (the edge). However, most of the machine learning and AI algorithms require a significant amount of processing power which may not always be available at the edge. We present an architecture that is based on a distributed edge/cloud paradigm and provides a good balance between the benefits and cost of processing data at the edge versus at a central location.

In Section II we describe some motivating examples for using IoT and AI together at the edge. A discussion of different approaches for applying AI to IoT data is presented in Section III, followed by a description of a system that can be used to ease the task of creating edge AI applications in Section IV. In Section V, we describe several ways in which policy technologies can be used in our system. Finally, in Section VI we state our conclusions and mention some future areas of study.

II. IoT applications requiring AI

IoT applications refer to any application monitoring and optionally manipulating a physical environment. Examples include tracking animals, surveillance and border patrol using drones, maintaining building equipment, airplane telemetry, manipulating blinds via voice commands, etc. Many of these applications require the use of sophisticated AI capabilities, including image, audio and video analysis.

A swarm of drones examining a remote forest fire or a collapsed building requires the capability to identify smoldering areas or people/pets that may be trapped. Each drone needs the ability to identify and detect objects, or areas that may have potential problems, like a smoldering fire. Their task is further complicated by an inability to connect and transmit large quantities of data over wireless networks or to receive instructions from a central controller in a timely fashion. These problems may also be exacerbated by the confusing terrain and rapidly changing circumstances typical of disaster environments. The overall system needs to be able to perform sophisticated object detection, identify the areas which may still be smoldering, and direct quenching mechanisms to the area.

Similarly, if we consider an agricultural vehicle tasked with the responsibility of spraying weed-killer over a large wheat farm, the vehicle needs to distinguish possible weeds from the desired crop, and spray the weed-killer only over the weeds while travelling at speeds of 50 miles per hour. This requires sophisticated image processing to distinguish weeds from crops in a sea of green. Furthermore, this identification and reaction needs to happen in less than 0.1 millisecond to achieve an accuracy of a centimeter or less.

Other applications include the ability of a system to listen to abnormal sounds from engines in a building equipment room and shut down malfunctioning motors, or the ability of a system to listen for drips/leaks from vats containing dangerous chemicals and initiating corrective action.

To attain the responsiveness required in these use-cases, data cannot be sent to the cloud for processing. However, the algorithms needed for good visual or audio recognition require a significant amount of compute power, more than can be expected to be present on a drone or a vehicle. Therefore, we need to create a hybrid architecture that can intelligently process data at appropriate points in the system.

III. Architectural alternatives for AI in IoT

AI approaches to address any problem can be abstracted in terms of the 2-stage process shown in Figure 1. In the first stage of the process, a set of AI models are built. These models can be built using machine learning algorithms with a set of training data, by processing natural language documents, or by the encoding of human expertise. Models can be expressed in different ways, e.g., as inference rule sets, decision trees, or neural networks. Once these models are built, they can be used

978-1-5386-2715-0/17/$31.00 ©2017 IEEE

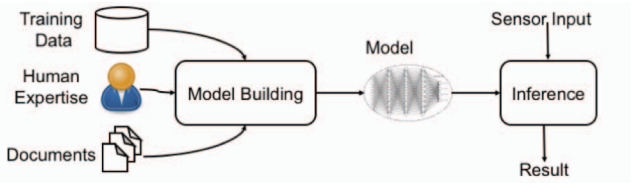

Figure 1. Abstracted AI approach

to make inferences from the sensor input data, and guide the operation of the system.

In the example of the agricultural vehicle, the training phase may consist of using a set of photographs of the desired crops and the undesired weeds to create a model that can distinguish between crops and weeds. In the inference stage, the vehicle uses the models to classify any crop it encounters and activate sprays of weed-killer for those that are undesirable. In the example of disaster recovery or fire prevention, the same kind of training can be done for models that can map images to hazardous situations. Similarly, recordings of normal and abnormal sounds of engines or leaks can be used to create models for equipment rooms or leakage of hazardous chemicals.

The model building step of an AI solution requires a significant amount of processing power. The best models are built with a large amount of training data, processing a large corpus of documents, or both. The required processing capacity is usually only available in a central location, a large data center or a cloud site. Centralization has other benefits – a central system running in a single homogenous environment is easier to maintain, program and upgrade. It has the benefit of global knowledge and visibility into all data that is available, and can provide better security, resiliency and reduce the per-transaction costs due to the economies of scale. The growth in popularity of cloud computing is largely due to the inherent advantages of centralized systems.

A natural way to build an AI enabled IoT system is to use a centralized approach. One can stream all the data collected from sensors in the field to a central location, and conduct both the model building and inference stages there. Using this approach, drones involved in disaster recovery would send all their information back to the central location for processing. The central location would have already created the required models, and would use them for performing the inference task as the images streamed in. If the network connection between the drone and the central location has sufficient bandwidth and a low latency, this approach can work effectively. This

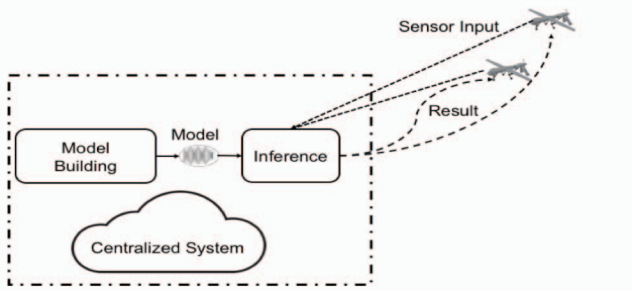

Figure 2. Centralized Approach

situation is shown in Figure 2. The centralized approach, however, may have certain disadvantages. In addition to the concern about latency, other considerations such as data privacy, data volume, or the cost of network connectivity must be taken into account. An alternative approach would be for the model building to be done in the cloud while the inference task is done at the edge, as in Figure 3.

The edge approach is bandwidth efficient and sends less data to the central location. In addition to reducing costs, and improving latency and responsiveness, this approach also has the advantage of improving the scalability of the system.

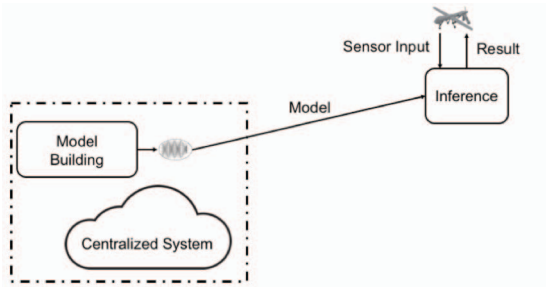

Figure 3. Edge Approach

Some AI paradigms are inherently distributed, an example being multi-agent systems [1]. Multi-Agent systems consist of distributed entities that communicate and interact with each other, basing their behavior on some learnt or preprogrammed model. However, such multi-agent systems interact with each other in complex ways, can have arbitrary topology, and can lead to instability and emergent phenomenon [2]. Such complexity is neither desirable nor necessary for AI enabled IoT solutions. A simpler approach, where the topology is essentially a tree structure with the edge devices as the leaf nodes of the tree and the centralized cloud system as the root of the tree, is sufficient to address a large range of practical IoT applications. Furthermore, just a tree with a depth of 1 consisting of only edge nodes and root nodes, is adequate for most IoT use-cases.

For most IoT applications, there is more involved than simply creating a model and using it at the edge. An application may consist of many different steps, some of which may be based on AI technologies and some may not. In general, we consider an application to be composed of several processing elements (PEs), some of which we refer to as Cognitive Processing Elements (CPE) because they use an AI model to produce their output. An application would then consist of a flow-chart of processing elements that could contain multiple branches. A good representation of such a workflow would be a produced by a system like Node-Red, which is a tool for building Internet of Things (IoT) applications with a focus on simplifying the 'wiring together' of code blocks to carry out tasks.

For ease if development, chains of processing elements are developed on the centralized environment, appropriately packaged along with any necessary supporting programs, and deployed to the edge. This allows the distributed processing of data streams by sophisticated analytics, while leveraging the power of the cloud environment for development of the

collections of processing elements needed, and overall management of the distributed edge environment.

IV. Prototype Edge Deployment System

Our long-term vision is to allow users to compose relevant applications on edge devices using cognitive components on the cloud with the simple push of a button. This approach leverages the computational resources of the cloud to: (a) perform the complex learning operation; (b) simultaneously model, test and rank a large number of algorithms before deciding on the appropriate analytics that will be deployed to the edge; (c) use micro-services that are widely available in the cloud to engineer new features that enrich the training data set and improve the selected model accuracy; and, (d) define a verifiable process for the composition and coordinated deployment of CPE chains to the edge devices.

While the availability of training data in the cloud is a key advantage of our proposed approach, assisting the data scientist in finding the relevant data is a primary concern for a cloud-based solution. Users should be able to upload their training data to the cloud after which edge programs are created automatically. They should also have access to labeled data that is made available by other scientists. This crowdsourcing approach to the dissemination and re-use of labeled data further enhances our ability to properly train and test the accuracy of the AI models in the cloud prior to their deployment to the edge. In some instances, it might be desirable to use simulated data generated in the cloud when real measurements from the edge devices are not available.

Templates for various CPE chains can be constructed in the cloud and associated with one or more AI models. The primary use of the CPE chain is to set up the data pipeline required for inferencing. A typical CPE chain includes five basic elements for performing data acquisition, feature engineering, aggregation, inferencing and reporting. CPE chains can also be augmented with user defined functions, such as user defined filters. They can also include system defined functions that are required for running the model on the edge, monitoring the performance of specific algorithms, and/or setting up triggers for reporting ML model scores or classification.

CPE chains can also be designed to include multiple cognitive elements sequenced together to form a more complex cognitive program where the classification output produced by one AI model can be used as an input feature in the feature vector of a second AI model.

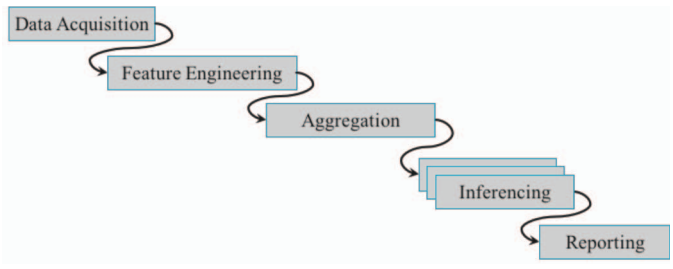

Figure 4. Typical elements of a CPE Chain

The overall capabilities can be described in terms of three phases: Discover, Deploy, and Operate. There is a fourth phase indicated in Figure 4 that is being explored and will be briefly described later.

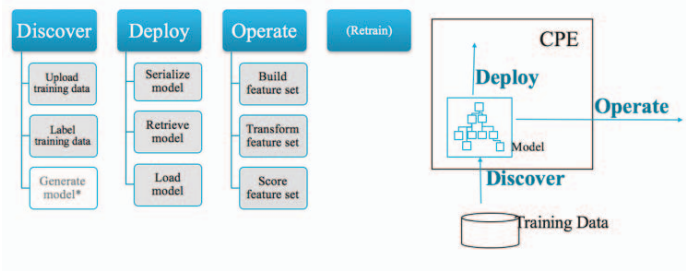

Figure 5. Services Supporting Key CPE Capabilities

In the Discover phase, a user will be uploading training data, labelling it, and testing multiple machine learning models. There are several libraries from which such models can be obtained. The idea is to discover the models that are most effective for the domain of the training data. The top N such models are identified based upon their associated FScore and Loss function. The user can then choose one or more models for deployment. Automated model generation uses a combination of techniques that consists of: (a) training multiple models simultaneously (i.e., SVN, Random Forest, K-NN, etc.); (b) applying known feature "engineering" methods to identify potential improvement on the selected model; and, (c) using test data to compare different implementations of the same model (Python vs Spark vs SPSS).

In the Deploy phase, the chosen models are retrieved, serialized, and packaged as Docker containers that are loaded into a shared repository from which they can be accessed by the edge components. The corresponding CPE chain is implemented as a Node-Red flow with each of the nodes within the flow implementing a wrapper around the processing elements identified in the CPE chain. The Node-Red flow also includes the initial word map constructed during the training phase and includes the word vocabulary of the training set.

In the Operate Phase, a micro-service that is running on the edge device is responsible for instantiating the CPE flow and executing the associated Docker container(s) on the edge device upon request by the user. Within the edge device, the CPEs are organized into CPE groups based on feature selections and transformations that are defined during the training phase. Since CPE groups have similar feature vectors, all the pre-requisite data transformations required for running the models in the group (i.e., feature engineering, aggregation, word hash, etc.) are done once for all the ML models in the group. The models are customized to the appropriate feature set, which can be extended to include aggregations and engineered features. The latter take advantage of additional information that may be available that is known to be effective in identifying the characteristics of interest.

The Retraining phase mentioned briefly above is meant to capture information about the effectiveness of the CPEs that

have been deployed so that they can be improved further. This feedback mechanism to the Cloud is meant to allow learning and continuous improvement.

Our method for model deployment as shown in Figure 6 is a framework that implements a set of REST APIs for automating the deployment of ML models to the edge. The System architecture is organized into a Cloud subsystem and an Edge Subsystem.

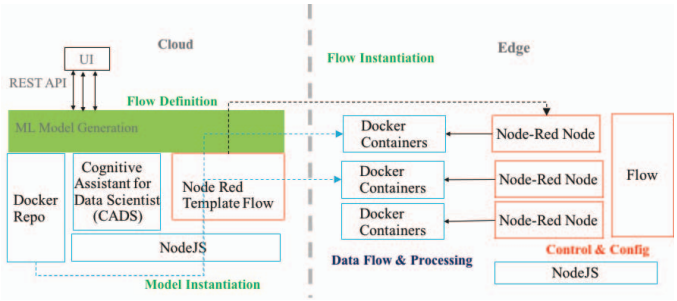

Figure 6. Model Deployment System Architecture

The Cloud subsystem implements the required services for composing the CPE flow, training the ML and exporting the trained model into a docker repository. A key component of the Cloud subsystem includes the Cognitive Assistant for Data Scientist (CADS) for training and comparing the inferencing performance of multiple ML models.

The Edge subsystem implements the required services for instantiating the CPE flow and corresponding docker containers representing the models referenced in the CPE flow.

V. POLICY TECHNOLOGIES

A. *Applications of policy technologies*

The application of policy technologies has proven to be very useful for simplifying the management and operations of complex distributed environments [3]. It allows decisions to be made as needed based on local context, while maintaining constraints imposed by the overall management system. Policies thus provide a way for software to be adapted to different edge environments, and for autonomous components to respond to changing conditions in acceptable ways. The Edge/Cloud model that has been described above increases the autonomy of the edge devices by allowing the independent processing of data streams by locally deployed analytics.

The management of the distributed environment must also preserve the autonomy of the edge components. The processing, communications, and storage overhead that would be inherent in any centralized approach to monitoring and maintaining detailed information about the local state of all the devices in the system, would be prohibitively costly. The use of policy technologies is thus a natural way of supporting distributed management. In our Edge/Cloud model, policies will be the key decision mechanisms in various management and operational processes.

Edge components will have management policies regarding their performance, the number of analyses they can perform, and the number of models that are relevant to the data in their local environment. Knowledge gained at one edge device could be communicated to other edge devices to coordinate activities and help them make choices. The applicability of such cross-component influences can be based on attributes of the various systems and their present context.

The more dynamic the interactions between the various edge components becomes, the more flexibility is needed in the underlying policy technologies. The current model for policy based management has the semantics that the policies set by the management system have fixed conditions and actions that are prescribed when the policies are deployed to the managed systems. A more recent paradigm for policy technologies enables the managed devices to generate their own policies [4]. These can change dynamically based on local context and direct interactions among the edge components, thus increasing the autonomy and responsiveness of the managed devices.

Policies can be used in various ways and to support different decision processes within the distributed system. They are not limited to operational management decisions. For example, given the number of different machine learning models, choices must be made concerning which models are most appropriate for which types of analysis and associated training sets. This information can be captured in policies, either reflecting existing knowledge or system experience.

B. *Coalition scenario with policy examples*

In coalition operations, e.g., when joint missions need to be undertaken by soldiers belonging to different countries, it is frequently essential to establish a dynamic community of interest. A dynamic community of interest (CoI) is a group of individuals that come together for a specified period to perform a mission, and the group dissolves after a specified time has elapsed or the mission has been completed. Such dynamic communities of interest may also be formed in non-military contexts, e.g., when different civilian agencies come together to fight a fire or deal with the aftermath of a hurricane. In a dynamic CoI, not all members are necessarily trusted equally, so policies related to information sharing may differ between groups of coalition members. It can be viewed as an instance of a sociotechnical system [5]

When dynamic CoIs are formed, they require supporting IT infrastructure to conduct their operations more effectively. The assets can come from all the different coalition members, and they need to interoperate. Software Defined Coalition (SDC) technologies provide such support [6]. They combine and extend concepts from software defined networking to operate in the context of coalition operations. A key technology needed to support software defined coalitions is a Policy Based Management System (PBMS). We will describe the types of policies that would be used within the context of a particular coalition scenario involving the assets of two nations, the US and the UK.

There has been an earthquake on an island. Many buildings have collapsed, people are trapped and the infrastructure is damaged. Several U.S. and UK ships are exercising in the area and offer assistance, but there will be a delay of a few hours before they arrive. Our interest in this scenario is in the deployment of drones, operated from the ships, that have been designed for search at sea. They need to be modified for search

in a land-based earthquake search mission, and this must be completed within a few hours.

These drones are equipped with cameras, and have (edge) inference engines for image processing to identify targets. Hence data scientists ashore start creating training data for the earthquake zone. This is a mix of synthetic (e.g., superimposing objects over bodies), open source information from other earthquakes, and a few images that arrive from first responders on the island who are overwhelmed and have minimal operational communications. This learning phase is carried out using U.S. facilities ashore and the outputs are sent to the U.S. and U.K. ships via satellite.

Many policies are applicable to this scenario. They can be used in the selection of training data from prior events to best match the characteristics of the island, e.g., foliage, and materials for building construction. Policies can also be used to select trusted sources of open source information. They can be applied to the selection of alternate sources of information, e.g., the use of human analysts to supplement the image processing on the drones, in a scenario where the performance of the drones may initially be poor but will improve following retraining. It may be required to preserve the privacy of certain images, for example to respect the sensitivity of VIPs who have property on the island.

The U.S. and UK drones have different capabilities, as do their sensors. Policies are used to make the best use of these resources, e.g., using some drones to identify areas likely to have casualties and others to locate casualties. Although the drones are designed to be autonomous (i.e., they do not require a remote pilot), there is an option for drones to be steered by forces on the ground with line of sight to obtain images in the complex damaged urban landscape. Policies include the assessment of the credentials of pilots and resource de-confliction if there is a high demand for the drones.

The machine learning algorithms themselves may make use of policies for selections of various alternatives. In an ensemble of different models, performance may be as important a consideration as the effectiveness of the models on different training sets. In a disaster situation, coming to a rough conclusion quickly is often more desirable than waiting for a detailed analysis. Similarly, one must weigh false positives (concluding that a person is trapped in the rubble of a building when that is not the case) against false negatives (failure to detect that a person is actually trapped). Generally, the effort to reduce one type of error results in increasing the other type of error. Policies can be used to favor one choice over the other based upon the operations being performed and the local context.

VI. Conclusions

We have presented an architecture for composing analytic packages on the cloud and deploying them for use at the edge. An infrastructure based on this architecture is being designed and prototyped. Some preliminary experiments have been very promising. Such a system allows processing of data streams to be distributed in a controlled manner, with intensive but well-structured processing being done at the edge. Edge computing can be very effective at reducing latency, minimizing data movement, and preserving local data privacy.

We continue to investigate the functionality split between Observers and Controllers (Edge and Cloud) and identifying the support and tooling required for efficiently composing distributed collaborative analysis applications for streaming data.

We have also shown how policy technologies can be used in the management and operational control of such dynamic, distributed systems. Examples have been given of the many ways in which policies naturally arise in the selection of processing alternatives and the management of system characteristics like performance, resiliency and privacy. Policy based mechanisms continue to evolve, given the requirements of IoT and collaborative computing. We are investigating ways to provide the necessary support for collaborative, autonomous systems, principally by the application of generative policy models. These allow managed systems to generate their own policies within the bounds imposed by the goals of the overall system of which they are a part. The effectiveness of such a paradigm, and the tools and data structures needed for its implementation are being studied.

Acknowledgment

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copy-right notation hereon. BAE and IBM have rights over the work produced by their employees.

References

[1] J. Ferber, Multi-Agent Systems: an introduction to distributed artificial intelligence, Addison-Wesley, 1999.

[2] J. Denzinger and J. Kidney, Evaluating different genetic operators in the testing for unwanted emergent behavior using evolutionary learning of behavior. In Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology, Dec 2006, pp. 23-29.

[3] S. Calo, D. Verma, E. Bertino, "Distributed Intelligence: Trends in the Management of Complex Systems", Proceedings of the 22nd ACM Symposium on Access Control Models and Technologies, SACMAT 2017.

[4] D. Verma, S. Calo, S. Chakraborty, E. Bertino, C. Williams, J. Tucker, Brian Rivera, Geeth R. de Mel, "Dynamic and adaptive policy models for coalition operations", Proc. SPIE, Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR VIII, Anaheim, California, USA, April 2017.

[5] M. Singh, Norms as a basis for governing sociotechnical systems. ACM Transactions on Intelligent Systems and Technology (TIST), *5*(1), 21, 2013

[6] Christopher Williams, Elisa Bertino, Seraphin Calo, Dinesh Verma, K. Leung, C. Dearlove, "Towards an Architecture for Policy-Based Management of Software Defined Coalitions", DAIS 2017 – Workshop on Distributed Analysis Infrastructure and Algorithms for Multi-Organization Federations, August 2017.