Towards Empowering Ubiquitous Computing as a Service with Cloud Analytics for Sustainable Manufacturing, Agriculture, Cities, and Buildings

1st Mirco Soderi
Data Science Institute
University of Galway
Galway, Ireland
mirco.soderi@universityofgalway.ie

2nd John Gerard Breslin
Data Science Institute
University of Galway
Galway, Ireland
john.breslin@universityofgalway.ie

***Abstract*—Cloud analytics extends the capabilities of identifying patterns, making predictions, and obtaining business intelligence (BI) insights, enabling organisations to work with massive amounts of complex business data using algorithms and cloud technologies. This type of analysis often involves artificial intelligence (AI), in the form of machine learning (ML) or deep learning (DL) models. Although concerns have been raised about the environmental impact of artificial intelligence and cloud computing, it is widely recognised that these technologies have the potential to support greater sustainability in several fields, including construction, transportation, healthcare, manufacturing, agriculture and water. This work extends our software framework for Ubiquitous Computing as a Service. The framework uses containerisation, low-code platforms, modularity, parallel computing, MQTT, and API to support the creation, configuration, operation, and modification of remote software. In this work, some degree of integration with cloud analytics is added, making it possible to create and populate AWS Redshift tables, train clustering models, and use such models, from containerised software components created and managed through the framework. All software artefacts produced in this work are available on GitHub, including a Postman collection of API requests for demo purposes.**

***Index Terms*—cloud analytics, big data, artificial intelligence, sustainability, smart building, transportation, healthcare, manufacturing, agriculture, water, software framework, ubiquitous computing, distributed system, containerisation, low-code, modularity, parallel computing, MQTT, API, clustering**

I. Introduction

Cloud analytics [1] extends the capabilities to identify patterns and make predictions, allowing organisations to work with large amounts of complex data using artificial intelligence (AI) [2] and cloud technologies. The application of AI to environmental sustainability presents a few challenges, including outdated data, cyber security threats, possible side effects, and lack of standardisation [3] [4]. However, a conspicuous literature stresses the potential of AI to face the great challenges of our times, especially when it extends to information systems [5]. AI can positively impact the five dimensions of sustainability [6], and in the literature encouraging applications that promote financial, health, and environmental inclusion in developing countries are described [7], along with efforts made for the protection and preservation of ecosystems [8]. However, there is still room for significant advances, in particular in energy efficiency, smart grid, and renewable energy [9].

Pervasive computing [10] [11], or ubiquitous computing, integrates connectivity functionalities into objects so that they can interact with each other and perform automated tasks with minimal human effort. Although concerns have been raised about the negative impact of pervasive computing on the environment and social sustainability, for example, in relation to power consumption, electronic waste, non-ionizing radiations, and unequal access to technology [12] [13], positive effects can still prevail if energy and waste are effectively governed in the coming years [14], and targeted training initiatives are undertaken [15]. Ubiquitous computing can help drive action and change in relation to environmental sustainability [16] and face the acceleration of the climate crisis, thanks to the development of new devices, services and tools [17], as demonstrated by its application to sustainable forest management [18] and ambient intelligence [19].

The synergy between pervasive computing and artificial intelligence has recently attracted growing interest. According to some authors, there cannot even be any pervasive computing without intelligent systems [20]. In fact, AI plays an important role in overcoming the challenges related to privacy, trust, and identity that are very present in ubiquitous computing [21], as well as in avoiding communication and computational overheads in IoT infrastructures [22]. Ambient intelligence has benefited significantly from pervasive AI [23], as well as the health sector [24] and smart cities [25].

This publication has emanated from research supported in part by the European Digital Innovation Hub Data2Sustain, co-funded by Ireland's National Recovery and Resilience Plan (the EU's Recovery and Resilience Facility), the Digital Europe Programme, and the Government of Ireland, and by a grant from Taighde Éireann - Research Ireland under Grant Number 12/RC/2289_P2 (Insight). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. A heartfelt thank you goes to the WiSAR Lab for logistic support, encouragement, and the lovely conversations.

II. Background

This work builds on our software framework for pervasive computing as a service. The entry point of the framework is the Network Factory [26], a containerised Node-RED application available on the Docker Hub as msoderi/network-factory, which exposes several APIs that allow one to create and organise containerised software locally or remotely, such as the Service Nodes [27].

Service nodes are configurable Node-RED applications that expose APIs to configure the task to execute, data sources and destinations, and more. The implementation is loaded from the Transformation Library, which contains a collection of Node-RED subflows including calculators, Artificial Intelligence Server (AIS) clients [28], real-time data visualisation components [29], Bluetooth Low Energy (BLE) servers [30], Postman collection runners [26], and more.

The Artificial Intelligence Server (AIS) is a Scala + Spark modular and containerised application that we have developed and that a Network Factory can instantiate. They expose APIs so that the Service Nodes that run the *ai* task/subflow can interface with them to configure, start, and stop the job execution. This can be done by multiple service nodes at the same time. The interaction between AIS and *ai* Service Nodes is shown in Fig. 1.

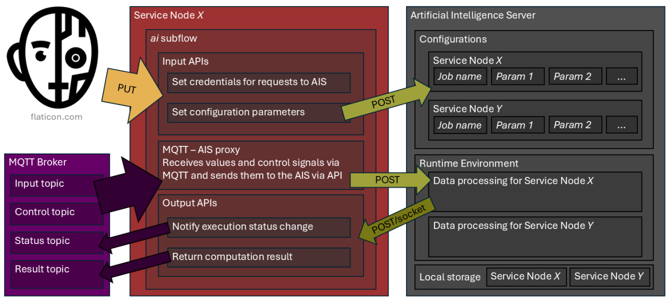


Fig. 1. Interaction between Service Nodes and AIS

For each client node, the server (i) maintains a separate configuration, (ii) allocates a portion of the file system, and (iii) asynchronously returns results and status information through API requests or through the socket. Servers include jobs that can take either the form of Scala classes (efficient), or plain text containing Scala code (less efficient, but can be added at run-time through API requests, and are compiled if and when needed). Servers come with some jobs already implemented for real-time data processing, as well as training of clustering, classification, and regression models, and usage of such models for predictions.

Remarkably, in Node-RED, it is possible to implement APIs that make arbitrary changes to the implementation of the application of which they are part, with immediate effect. We use this feature in Crazy Nodes [31], which are specialised Service Nodes. This helps in building automated or semi-automated systems that are resilient to both predictable and unpredictable events.

III. Cloud Analytics Integration

Three new functions (Node-RED subflows) have been introduced in the Transformation Library to support cloud analytics by (i) populating training datasets; (ii) creating models; (iii) obtaining predictions. Although fairly specific, this improvement paves the way towards deeper integration.

A. The awsrstbli subflow

The awsrstbli subflow (Fig. 2) is used to populate a table in AWS RedShift with numeric data.

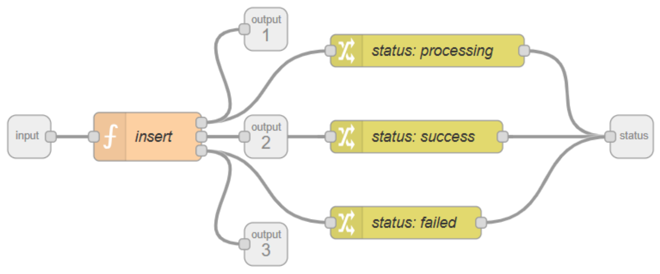


Fig. 2. The awsrstbli subflow

It exposes a few APIs for the configuration of the region, access key ID, secret access key, workgroup name, database name, table name, and numeric data type.

It receives numeric data through the *input* node, then it interfaces to AWS Redshift in the *insert* node, outputting to *output 1* when a query is accepted, to *output 2* when a query completes successfully, and to *output 3* when a query fails. It also outputs to the *status* node messages with an appropriate payload, as described in the Node-RED documentation, so that if you are looking at the subflow node in operation through the Node-RED Web interface, you can have some kind of visual feedback on how things are going.

In the *insert* node, the AWS SDK for JavaScript is used, and the following happens for each message received in input: (i) a connection is opened to AWS Redshift; (ii) a table is created with a predefined schema and the configured name if not already present; then a new row is inserted into the table that carries the value received in input, with the two queries wrapped in a single `execute statement` command; (iii) the response to the command submission is outputted to *output 1*; (iv) a wise polling takes place to determine when the query execution completes, which is done through `describe statement` commands; (v) the response to the `describe` command is outputted to *output 2* (success) or to *output 3* (fail) when the *status* reported in the response is either *FINISHED*, or *FAILED* respectively.

B. The awsrskmc subflow

The awsrskmc subflow (Fig. 3) is used for training and storing K-Means models in AWS Redshift.

It exposes configuration APIs that are used to set the access key ID and secret access key. Most importantly, it exposes a *model* API that is expected to be called with the following

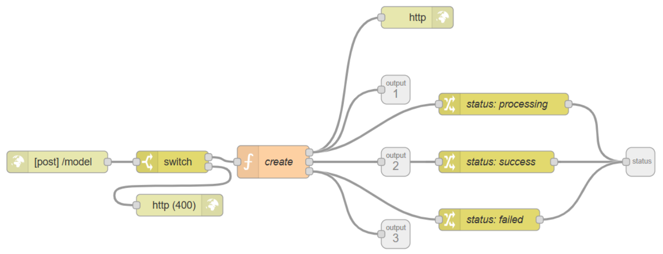


Fig. 3. The awsrskmc subflow

properties set in the JSON payload: (i) region; (ii) workgroup name; (iii) database name; (iv) table name; (v) model name; (vi) function name; (vii) K, the K-Means parameter; (viii) S3 bucket name. The *switch* node checks the validity of the payload. In the *create* node, the following occurs for each message received in input, with the support of the AWS SDK for JavaScript: (i) the connection to AWS Redshift is established; (ii) a `execute statement` command is issued to run the Redshift query that trains the model using the data present in the specified table, then stores the trained model in the specified S3 bucket; (iii) the same steps described in Section III-A are followed.

C. The awsrskmp subflow

The *awsrskmp* subflow (Fig. 4) is used to get AWS Redshift predictions. For each value that comes from the *input* node, it uses the configured existing K-Means model to determine to which cluster the value belongs.

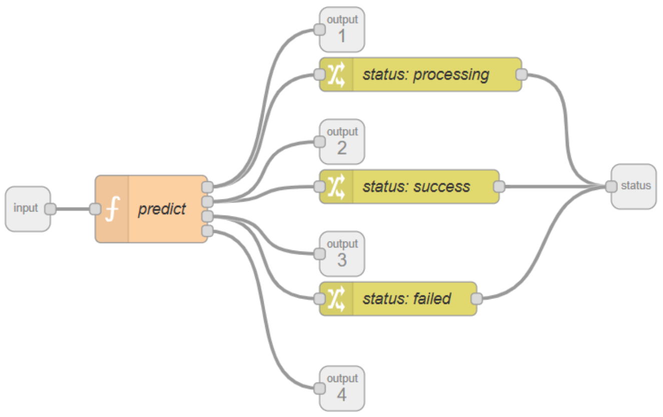


Fig. 4. The awsrskmp subflow

When you train and store a model in AWS Redshift, as a part of the query that creates the model, you specify a function that you will use later in queries to obtain predictions from the model you are creating. For using this subflow, you first need to use the configuration APIs that it exposes, and set (i) region, (ii) access key ID, (iii) secret access key, (iv) workgroup name, (v) database name, (vi) table name, and (vii) the name of the function that is associated with the model to be used.

At that point, in the *predict* node, for each value received in input, a connection to AWS Redshift is opened, and a query is sent to retrieve the full list of the clusters that are present in the selected model, with the following information included for each cluster: (i) a numeric unique identifier; (ii) the size, which is the count of the training data points that belong to that cluster; (iii) the highest and lowest of the training data points that belong to the cluster; (iv) a boolean flag that indicates whether or not the value received in input belongs to the specific cluster.

Remarkably, this subflow has four outputs, instead of three, because the first three are used for the same purposes described above for the other subflows, but then we need a fourth output that is used to output the result of the query; indeed, the `describe` command can be used to know the execution status of a query, but does not return the result, which is returned by a separate command, namely the `get statement result` command.

D. Main challenges

A significant challenge was understanding the authentication and authorisation mechanisms, and the way how security is managed in the specific cloud system under consideration, so that it could be possible to interact with it from the external conforming to best practices.

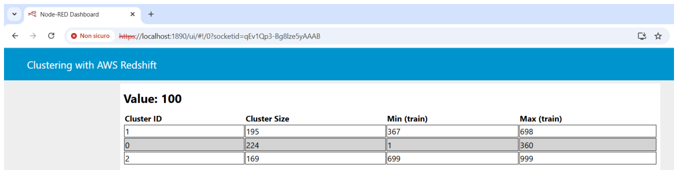
A second significant challenge was managing the inherently asynchronous nature of the requests sent to the data warehouse, which required deciding which were the relevant events, how to efficiently subscribe or poll, and how to notify the user that some relevant event has occurred.

IV. Evaluation

This work has been evaluated by creating a system that includes: (i) the *InjectAwsrstbiSN* service node, which exposes an API that is used for inputting training values and is connected via the MQTT topic redshift/training to the *AwsrstbliSN* service node, which is responsible for creating the Redshift table if necessary, then adding a row with the value received via MQTT; (ii) the *AwsrskmcSN* service node, which exposes an API that is used for creating the K-Means model in Redshift, plus configuration APIs that are used for setting the credentials to be used when interfacing with Redshift; (iii) the *InjectAwsrskmpSN* service node, which exposes an API that is used for inputting the values about which we need a prediction and is connected through the MQTT topic redshift/input to the *AwsrskmpSN* service node that provides the prediction and sends it over via the MQTT topic redshift/prediction to the *CreateHtmlSN* service node that creates the HTML markup out of the JSON response generated by *AwsrskmpSN* and sends such markup over via the MQTT topic redshift/html to the *DisplayHtmlSN* service node, which finally renders the markup on a web page (Fig. 5).

The system is created through a Postman collection of API requests[1] organised into three top-level folders: (i) *Build*, where the requests for creating the system locate; (ii) *Use*,

[1]https://github.com/mircosoderi/State-of-the-art-Artefacts-for-Big-Data-Engineering-and-Analytics-as-a-Service/blob/main/BDEAaaS-Redshift.postman_collection


Node-RED Dashboard
Non sicuro
~~https~~://localhost:1890/ui/#!/0?socketid=qEv1Qp3-Bg8lze5yAAAB
Clustering with AWS Redshift

Value: 100

Cluster ID	Cluster Size	Min (train)	Max (train)
1	195	367	698
0	224	1	360
2	169	699	999

Fig. 5. Demo

where the requests to watch the system in operation locate; (iii) *Dismiss*, where the requests to clean up the Docker environment are located.

In terms of execution times, requests that are aimed at creating the Service Nodes, the broker, and the transformation library tend to be slower because they encompass the download of software artefacts from the GitHub repository and possibly the download of Docker images from the Docker Hub.

V. Conclusion

This work constitutes an example of cost-effective integration of reconfigurable ubiquitous computing with cloud technologies made available by third-party organisations.

This work can find applications in manufacturing to monitor sensor values and raise alerts as appropriate, on scale. The alert can then flow to other parts of the system built using our framework to achieve automated resiliency and build human-friendly Web interfaces.

In the future, integration with multiple and diverse cloud services can be provided through the implementation of additional subflows in the transformation library.

References

[1] B. Berisha, E. Mëziu, and I. Shabani, "Big data analytics in cloud computing: an overview," *Journal of Cloud Computing*, vol. 11, no. 1, p. 24, 2022.

[2] P. H. Winston, *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1984.

[3] R. Nishant, M. Kennedy, and J. Corbett, "Artificial intelligence for sustainability: Challenges, opportunities, and a research agenda," *International Journal of Information Management*, vol. 53, p. 102104, 2020.

[4] N. A. Rakha, "Artificial intelligence and sustainability," *International Journal of Cyber Law*, vol. 1, no. 3, 2023.

[5] T. Schoormann, G. Strobel, F. Möller, D. Petrik, and P. Zschech, "Artificial intelligence for sustainability—a systematic review of information systems literature," *Communications of the Association for Information Systems*, vol. 52, no. 1, p. 8, 2023.

[6] J. Khakurel, B. Penzenstadler, J. Porras, A. Knutas, and W. Zhang, "The rise of artificial intelligence under the lens of sustainability," *Technologies*, vol. 6, no. 4, p. 100, 2018.

[7] B. A. Ojokoh, O. W. Samuel, O. M. Omisore, O. A. Sarumi, P. A. Idowu, E. R. Chimusa, A. Darwish, A. F. Adekoya, and F. A. Katsriku, "Big data, analytics and artificial intelligence for sustainability," p. e00551, 2020.

[8] P. Dauvergne, *AI in the Wild: Sustainability in the Age of Artificial Intelligence*. MIT Press, 2020.

[9] N. Bracarense, R. E. Bawack, S. Fosso Wamba, and K. D. A. Carillo, "Artificial intelligence and sustainability: A bibliometric analysis and future research directions," *Pacific Asia Journal of the Association for Information Systems*, vol. 14, no. 2, p. 9, 2022.

[10] D. Saha, A. Mukherjee, S. Bandyopadhyay, D. Saha, A. Mukherjee, and S. Bandyopadhyay, "Pervasive computing," *Networking Infrastructure for Pervasive Computing: Enabling Technologies and Systems*, pp. 1–37, 2003.

[11] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Personal communications*, vol. 8, no. 4, pp. 10–17, 2001.

[12] A. Koehler and C. Som, "Effects of pervasive computing on sustainable," *IEEE Technology and Society Magazine*, vol. 24, no. 1, pp. 15–23, 2005.

[13] L. M. Hilty, C. Som, and A. Köhler, "Assessing the human, social, and environmental risks of pervasive computing," *Human and Ecological Risk Assessment*, vol. 10, no. 5, pp. 853–874, 2004.

[14] A. Köhler and L. Erdmann, "Expected environmental impacts of pervasive computing," *Human and Ecological Risk Assessment*, vol. 10, no. 5, pp. 831–852, 2004.

[15] J. Porras, A. Seffah, E. Rondeau, K. Andersson, and A. Klimova, "Perccom: A master program in pervasive computing and communications for sustainable development," in *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*. IEEE, 2016, pp. 204–212.

[16] M. Foth, E. Paulos, C. Satchell, and P. Dourish, "Pervasive computing and environmental sustainability: Two conference workshops," *IEEE Pervasive Computing*, vol. 8, no. 1, pp. 78–81, 2008.

[17] M. L. Mauriello and M. Hazas, "Pervasive sustainability," *IEEE Pervasive Computing*, vol. 23, no. 2, pp. 4–6, 2024.

[18] M. F. Khan, "Pervasive computing for sustainable forestry management: Developing an iot-connected timber harvesting and forest conservation platform."

[19] A. Bimpas, J. Violos, A. Leivadeas, and I. Varlamis, "Leveraging pervasive computing for ambient intelligence: A survey on recent advancements, applications and open challenges," *Computer Networks*, vol. 239, p. 110156, 2024.

[20] S. G. Thompson and B. Azvine, "No pervasive computing without intelligent systems," *BT technology journal*, vol. 22, no. 3, pp. 39–49, 2004.

[21] G. DAngelo, S. Rampone, and F. Palmieri, "An artificial intelligence-based trust model for pervasive computing," in *2015 10th international conference on P2p, parallel, grid, cloud and internet computing (3pgcic)*. IEEE, 2015, pp. 701–706.

[22] E. Baccour, N. Mhaisen, A. A. Abdellatif, A. Erbad, A. Mohamed, M. Hamdi, and M. Guizani, "Pervasive ai for iot applications: A survey on resource-efficient distributed artificial intelligence," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2366–2418, 2022.

[23] J. C. Augusto, "Ambient intelligence: the confluence of ubiquitous/pervasive computing and artificial intelligence," in *Intelligent computing everywhere*. Springer, 2007, pp. 213–234.

[24] N. VR and R. PS, "Pervasive computing in the context of covid-19 prediction with ai-based algorithms," *International Journal of Pervasive Computing and Communications*, vol. 16, no. 5, pp. 477–487, 2020.

[25] J. Nigon, E. Glize, D. Dupas, F. Crasnier, and J. Boes, "Use cases of pervasive artificial intelligence for smart cities challenges," in *2016 Intl IEEE conferences on ubiquitous intelligence & computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people, and smart world congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE, 2016, pp. 1021–1027.

[26] M. Soderi and J. G. Breslin, "A service for resilient manufacturing," in *2023 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2023, pp. 195–197.

[27] M. Soderi, V. Kamath, J. Morgan, and J. G. Breslin, "Ubiquitous system integration as a service in smart factories," in *2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*. IEEE, 2021, pp. 261–267.

[28] M. Soderi, V. Kamath, and J. G. Breslin, "A demo of a software platform for ubiquitous big data engineering, visualization, and analytics, via reconfigurable micro-services, in smart factories," in *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2022, pp. 1–3.

[29] ——, "Toward an api-driven infinite cyber-screen for custom real-time display of big data streams," in *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2022, pp. 153–155.

[30] M. Soderi and J. Gerard, "Ble servers and ubiquitous analytics aas," *AICS 2022 Digital Book of Abstracts*, 2022.

[31] M. Soderi and J. G. Breslin, "Crazy nodes: towards ultimate flexibility in ubiquitous big data stream engineering, visualisation, and analytics, in smart factories," in *International Symposium on Leveraging Applications of Formal Methods*. Springer, 2022, pp. 235–240.