Decentralized Microservice Workflows for Coalition Environments

Chris Simpkin and Ian Taylor
Cardiff University
simpkin.chris@gmail.com
taylorij1@cardiff.cf.ac.uk

Graham A Bent and Geeth De Mel
IBM UK
{GBent,geeth.demel}@uk.ibm.com

Swati Rallapalli
IBM US
srallapalli@us.ibm.com

The workflow methodology provides a robust means of describing applications consisting of control and data dependencies along with the logical reasoning necessary for distributed execution. For wired networks, there have been a wide variety of successful workflow systems available for researchers to design, test and run scientific workflows [17], [6], [1], [12], [4], [14], [10], [3], [7], [2]. Similarly, in the business domain, the Workflow Management Coalition (WfMC) [19] has delivered standards to a vast number of business workflow systems for over twenty years. It is important to note that these workflows require centralized management, thus applying such in dynamic coalition network environments raises several technical challenges, for example: *(a)* variable network connectivity associated with mobile endpoints (e.g., unmanned autonomous systems); *(b)* high latency and cost associated with communication (e.g., satellite); and *(c)* poor infrastructure, especially absent back-end connectivity .

Consequently, in dynamic environments, a new class of workflow methodology is required—i.e., a workflow which operates in a decentralized manner. In the extreme case, the system should be capable of orchestrating a workflow with complete decentralization. The workflow execution logic dictating the flow should be capable of using local knowledge or knowledge collected from prior distributed communications wherever possible. The work in this paper aims at investigating such decentralized approaches by focusing on common methodology to describe and orchestrate decentralized services to support mission critical data analytics pipelines.

Motivated by this observation, in this work, we envision workflows which are composed of microservices—i.e., services which move away from the more traditional large monolithic back-end applications by splitting the functionality into a set of smaller more manageable services. Typically, microservices communicate with each other using lightweight protocols (e.g., HTTP, REST) and are widely adopted in the industry by companies such as Netflix and Amazon with a large number of developers. However, such services are are designed as request/response services, which are therefore centralized in nature. However, in the scenarios being considered in this work, these services may belong to different coalition partners but a common goal needs to be achieved by composing them dynamically which yields to the need of service discovery.

Service discovery is a hard problem even when services are hosted in centralized repositories. This is mainly because microservices are often developed and deployed independently or by loosely cooperating developers in open environments. This has led to a complex mix of disparate microservice architectures employing different methodologies for the description of their input, outputs, configuration—and in coalition environments their policy requirements. Thus, there is no standardized methodology for service description.

In recent years, especially in support of services API economy, where monetization for API developers is the focus, social tagging of services are considered as means to reduce the burden on developers to annotate services with full descriptions; once captured and stored in repositories, these social tags can be used to assist service discovery in matching contexts. For example, in [18] multiple parties would tag services based on the context in which they have used services. These annotations are then used by a learning algorithm to augment a knowledge base—a one based on resource description framework (RDF) [15]—so that an evolving graph of interactions is captured to infer potential uses of services with respect to annotations provided by the past users and the new requirements post by the current users.

In distributed setting, the above approach is impractical, if not unusable. This is because in services API economy, the services are centrally hosted, thus the hosting system has a bird's-eye-view of the registered services which enables it to easily monitor service usage patterns, thus making the annotation learning and correlation straightforward. However, in distributed service environment, having such a global view is impossible.

In support of such situations, we have investigated techniques that will allow microservices to be deployed independently onto a MANET using disparate service descriptions while enabling them to be discovered by service requesters and allowing themselves to learn what workflows they can participate in. In addition, we show how multiple copies of the same microservice can be distributed throughout the MANET, and describe a methodology that will enable matching services to offer themselves up based on best utility, in terms of cost of transaction with respect to available processing power, cost of data access and, in coalition environments, security policy.

Vector Symbolic Architectures (VSAs) are a set of lossy dimensionality reduction methodologies that enable large volumes of data to be compressed into a fixed size data block (vector) in a way that captures associations and similarities as well as enabling categorizations between data to be built up. The two basic operations that enable these capabilities are superposition (addition) and binding (multiplication/permutation). Superposition builds up similarities between vectors whereas binding combines vectors into the same space in an orthogonal manner. Such vector representations are recursive as originally proposed by Hinton [11] in that they allow for higher level abstractions to be formulated in the same format as their lower level components. Plate's Holographic Reduced Representation (HRR) [18] describes how to use such operations on vector symbols, via role-filler, pairs in order to maintain positional or temporal relationships between objects as well as to learn deep and shallow semantic relationships between objects. In [13] Kanerva describes the use of Binary Spatter Codes (BSC) in combination with Random Permutations (RPM) to achieve a computationally more efficient version of the same, see [16] for a comparison. In addition word2Vec[8] and node2Vec[9] might offer up methodologies that can learn microservice descriptions and iteratively traverse and learn a distributed microservice graph.

In the context of a workflow, a high level vector may represent an action such as 'Track Person'. This vector would contain within itself the sub-component workflows, 'get CTTV video streams' + 'search for person' + 'Predict path'. Each of which would also recursively resolve to the individual microservices needed to complete the high level request. The key point is that, in order for this service composition to be activated on the MANET, only the single vector symbol 'Track Person' would be broadcast over the network since this single vector contains a full description of the microservices needed to execute the request and the graph of how such microservices must to be connected. Each microservice would examine the vector and activate itself in the appropriate position for its part in the workflow.

The fact that vector pointers can be combined into a fixed size reduced representation and that learning can be carried out iteratively is highly attractive for employment in a distributed learning environment. In addition, the fact that higher level representations have the same size and form as their lower level components offers the potential for service composition requests to be made in more intuitive and human natural language. Further, the fixed size nature of VSAs are ideally suited for implementation in non-von Neumann neuromorphic technology which will be highly advantageous for mobile devices operating on the edge due to the multiple orders of magnitude reduction in power consumption of such devices. For example, Eliasmith uses HRRs to implement a biologically plausible cognitive computer in spiking neural networks [5].

We believe these methodologies could be used and extended to learn similarities between microservices based on their vector description (input, output, policy, natural language description) which will allow them to initially start to participate in workflows. Best fit will be further reinforced when each service, as well as service requesters, learn the historical contexts and positions in which a microservice has been invoked. Each microservice is effectively acting as an associative clean up memory. This will enable alternate best-fit functionality to be achieved for robust operations in the frequently changing and noisy environments of coalition MANETS.

References

[1] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock. Kepler: An Extensible System for Design and Execution of Scientific Workflows. In *16th International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 423–424. IEEE Computer Society, New York, 2004.

[2] B. Balis. Increasing scientific workflow programming productivity with hyperflow. In *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science*, WORKS '14, pages 59–69, Piscataway, NJ, USA, 2014. IEEE Press.

[3] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan. The trident scientific workflow workbench. In *Proceedings of the 2008 Fourth IEEE International Conference on eScience*, pages 317–318, Washington, DC, USA, 2008. IEEE Computer Society.

[4] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. Katz. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific Programming Journal*, 13(3):219–237, 2005.

[5] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen. A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205, Nov. 2012.

[6] T. Fahringer, R. Prodan, R.Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazon, and M. Wieczorek. *Workflows for e-Science*, chapter ASKALON: A Development and Grid Computing Environment for Scientific Workflows, pages 143–166. Springer, New York, 2007.

[7] T. Glatard, J. Montagnat, D. Lingrand, and X. Pennec. Flexible and efficient workflow deployment of data-intensive applications on grids with MOTEUR. *Int. J. High Perform. Comput. Appl.*, 22:347–360, August 2008.

[8] Y. Goldberg and O. Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[9] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.

[10] A. Harrison, I. Taylor, I. Wang, and M. Shields. WS-RF Workflow in Triana. *International Journal of High Performance Computing Applications*, 22(3):268–283, Aug. 2008.

[11] G. E. Hinton. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75, 1990.

[12] P. Kacsuk. P-grade portal family for grid infrastructures. *Concurr. Comput. : Pract. Exper.*, 23:235–245, March 2011.

[13] P. Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.

[14] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics*, 20(17):3045–3054, November 2004.

[15] J. Z. Pan. Resource description framework. In *Handbook on Ontologies*, pages 71–90. Springer, 2009.

[16] G. Recchia, M. Sahlgren, P. Kanerva, and M. N. Jones. Encoding sequential information in semantic space models: comparing holographic reduced representation and random permutation. *Computational intelligence and neuroscience*, 2015:58, 2015.

[17] M. Wieczorek, R. Prodan, and T. Fahringer. Scheduling of scientific workflows in the askalon grid environment. *SIGMOD Record*, 34(3):56–62, 2005.

[18] E. Wittern, J. Laredo, M. Vukovic, V. Muthusamy, and A. Slominski. A graph-based data model for api ecosystem insights. In *Web Services (ICWS), 2014 IEEE International Conference on*, pages 41–48. IEEE, 2014.

[19] The Workflow Management Coalition. http://www.wfmc.org/.