Emergent Scheduling of Distributed Execution Frameworks

Paul Allan Dean
Supervisor: Dr. Barry Porter
School of computing and Communication
Lancaster University
Lancaster, UK
(p.dean1, b.f.porter)@lancaster.ac.uk

I. Motivation & Problem

Distributed Execution Frameworks (DEFs) provide a platform for handling the increasing volume of data available to distributed computational processes, forming the creation and usage of a large number of DEFs for performing distributed computations. For example, sorting and analyzing large data sets through map and reduce operations [1], performing a set of operations across points in a data stream to provide near real-time analysis [2],and the training and testing of machine learning models for varying methods of learning, such as, supervised, unsupervised and reinforcement learning [3], [4], exploiting the vast amounts of data available. However, this has led to various DEFs becoming optimal for either fine or course-grained computations. Prominent examples for course-grained data-parallel DEFs are Apache Hadoop, utilising the MapReduce programming model [5], and Apache Spark which offered a significant performance increase over Hadoop for iterative workloads while hindering the capabilities of handling fine-grained tasks due to scheduling latency [1]. Whereas, Apache Flink ,offers real-time processing of workloads as data streams, while sacrificing the locality aware data-parallelism offered by Spark [2].

Furthermore, Ray also highlights the issues presented by Sparks inability to handle latency sensitive tasks and the ever growing types of workloads available and offers a low-latency DEF while again similar to Flink, Ray sacrifices data-locality hindering performance for data-parallel computation [4]

The increasing number of varying frameworks, designed for different workloads, illustrate the problem DEFs are facing with meeting the changing computational needs of organisations and current DEFs inability to adapt to a new set of requirements.

Therefore, this PhD will focus on overcoming the issue of trading performance for differing workloads by exploiting the capabilities presented by emergent software systems which learn how to assemble and re-assemble themselves in response to their current deployment conditions and input pattern [6]. This allows the creation of a component based DEF capable of altering both the local behaviour of a DEF (i.e. Local Schedulers and placement polices within a centralised scheduler) to potentially improve the performance of single DEF as well as global behaviour of a DEF, for example the adaptation of a centralised to two-level scheduler.

The creation of an emergent DEF presents the problem of identifying points during the execution of a workload which would signify when a change in behaviour is required to improve a specific performance metric (E.g. Job Completion Time). Therefore, the projects key problem to be addressed is the exploration of a learning agent capable of learning, identifying and selecting the optimal composition for master and worker nodes at runtime. The problem consist of several issues which are to be addressed during the project, (i) finding an agent capable of exploring the given search space of available compositions and identifying the optimal usage of each. (ii) The workloads used to explore the performance of compositions are stochastic, often non-repeating and their arrival sporadic, as such a method for overcoming the presented challenge will be investigated. (iii) The learning process is required across all nodes within a given cluster and each node may present differing types and quantity of tasks dependent on the global scheduler. All of which is to be performed for each DEF framework available and presents the problem of avoiding the need to re-learn previously explored components/compositions performance for a given workload/tasks. Thus emphasizing the need for the transfer of learning between DEFs to avoid the costly process of re-training the agent [7].

II. Contribution & Objectives

The PhD intends to deliver two contributions, the first, a Emergent DEF capable of swapping components at runtime and allowing the optimal composition of components for a given workload to be assembled during runtime. Which would produce a DEF capable of avoiding the inherently fixed nature of previously discussed DEFs. Second, a learning agent capable of learning within a distributed environment and transferring the previously learned compositions and their performance to an altered set of components and available compositions. This is particularly important in the context of DEFs due to the sporadic, non-repetitive nature of workloads, which makes typical reinforcement learning approaches difficult to use.

III. Preliminary Results & Methodology

The rationale behind the Initial experiments is to understand whether or not divergent optimal compositions exist for common kinds of DEF workload. Analysis of preliminary results signifies points where a learning agent may adapt compositions for improved performance and forms the motivation for the PhD's research into an emergent solution. Experiments will be performed across 5 nodes, 2 workers comprised of a singe Intel Xeon Processor E3-1280 v2 with 4 cores (8Threads) and 16GB memory with an additional 2 workers and master node consisting of a Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz with 8 cores (16 threads) and 16GB memory. The experiments will measure the total job completion time for a three workloads for three differing cluster resource schedulers, FIFO, Naive Fair and Dominant Resource Fairness [8], [9]. Each workload consists of 20 applications with differing percentages of fine and course granularity tasks. Workload 1 consists of 80% course grained applications and remainder fair. workload 2 is 80% fine grained and 20% course creating a heavy-tailed distribution (large portion of work is in a small percentage of the workload). Finally the third consist of 50% fine and course. Resource requirements for course grained applications vary between 2-4 cores and 2-6 Gigabytes of memory. Whereas fine-grained tasks are between 1-2 cores and 1-2 Gigabytes of memory. Each workload represents a set of tasks representing common workloads encountered by various previously discussed DEFs.

IV. Future Work & Research Plan

Future work consist of the completion of the emergent DEF and the creation of a learning agent to address the previously discussed problems.

Completion of the DEF entails adding more behaviours with significant differences (E.g. real-time data processing of data streams & Machine Learning model training), providing additional compositions to address a more diverse set of workloads building upon the current set of components for common DEF workloads. In addition, to exploring potential learning agents for assessing the capabilities of compilations, addressing the first of the previously discussed problems.

Initial exploration will investigate the use of Reinforcement Learning agent and techniques for handling the previously mentioned characteristics of workloads which create difficulties in learning optimal compositions, with the investigation of variance reduction techniques [10], presenting improvements in the training of a reinforcement learning agent using input sequences which may be non-repeatable. Therefore, providing an initial avenue of exploration for addressing the second previously mentioned problem.

Subsequently leading to the final avenue of future work exploring the application of transfer learning techniques to address the problem of a changing state and action space across DEFs, with the motivation of addressing the third stated problem of reducing the need to re-learn previously explored assembled compositions.

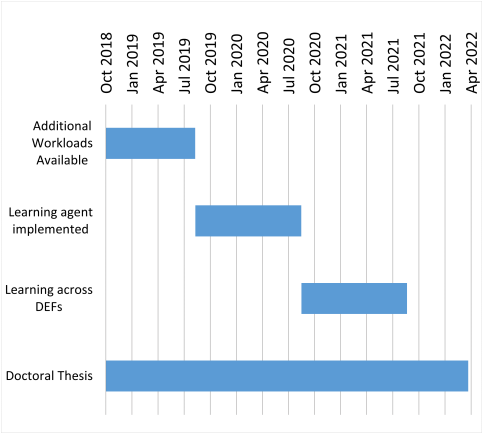

Fig. 1. PhD research plan

A. Research Plan

The following section summarizes milestones within the PhD research plan, with each milestone progressing the PhD project and finally aiding in forming the doctoral Thesis (Figure 1).

August 31, 2019
: Additional components addressing a more diverse set of workloads, for example Unsupervised Learning and Stream based computation. Providing a DEF a learning agent may exploit and re-assemble

August 31, 2020
: A Learning agent has been selection from the initial exploration and has been implemented, trained and tested across a single DEF addressing the challenges of learning from given workloads

August 31, 2021
: Learning across all Distributed Execution Frameworks has been addressed through the exploration and implementation of a transfer learning technique to provide a generalized learning algorithm for the problem space.

March 30, 2022
: Completed Doctoral Thesis

Acknowledgment

This work was partially funded by a PhD studentship from Lancaster university Faculty of Science and Technology, and by the UK Leverhulme Trust via the *Self-Aware Datacentre* project, grant RPG-2017-166.

REFERENCES

[1] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863103.1863113

[2] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *IEEE Data Eng. Bull.*, vol. 38, pp. 28–38, 2015.

[3] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "Mllib: Machine learning in apache spark," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1235–1241, Jan. 2016. [Online]. Available: http://dl.acm.org/citation.cfm?id=2946645.2946679

[4] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging AI applications," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, 2018, pp. 561–577. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/moritz

[5] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, 2004, pp. 137–150.

[6] R. R. Filho and B. Porter, "Defining emergent software using continuous self-assembly, perception, and learning," *ACM Trans. Auton. Adapt. Syst.*, vol. 12, no. 3, pp. 16:1–16:25, Sep. 2017. [Online]. Available: http://doi.acm.org/10.1145/3092691

[7] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Dec. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1577069.1755839

[8] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proceedings of the 5th European Conference on Computer Systems*, ser. EuroSys '10. New York, NY, USA: ACM, 2010, pp. 265–278. [Online]. Available: http://doi.acm.org/10.1145/1755913.1755940

[9] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 323–336. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972490

[10] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, "Variance reduction for reinforcement learning in input-driven environments," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Hyg1G2AqtQ