Towards Emergent Scheduling for Distributed Execution Frameworks

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

I. Abstract

The increased volume of data available to organisations, the computational resources available within data centres, and the demand for systems capable of processing Terabytes/Petabytes of data has influenced the development of numerous Distributed Execution Frameworks (DEFs), such as Apache Spark and Ray [1], [2].

DEFs have become a ubiquitous part of modern data processing systems, both as standalone processing frameworks and in a pipeline for a larger service, for example, feature extraction within a recommender system processing large volumes of data.

The wide range of workload types for DEFs make it extremely challenging to build a scheduler able to maximise throughput for all cases, causing scheduling overheads for certain workload types, for example, Apache Spark adds latency to all scheduling decisions hindering the performance of latency-sensitive workloads. In turn, this has prompted the creation of DEFs with differing schedulers for Reinforcement learning [2], Machine Learning (Deep Learning) [3], [4], and real-time stream processing Workloads [5]–[7]. However, each of these DEFs has become highly specialised to a particular workload type, for example Ray focuses on being able to efficiently schedule Reinforcement learning workloads [2]. Other DEF schedulers have been developed to improve scalability [8]–[11], while limiting the scheduling policy to a single metric and losing guarantees for data-locality, significantly hindering the performance of data-intensive operations. Hybrid DEFs attempted to benefit from two scheduling architectures providing efficient scheduling for a larger set of workloads [11], at the cost of being unable to exploit the full performance of a single architecture.

Previous approaches to adapting the schedulers of DEFs have been limited by their need for user intervention: policy adaptation requires user-provided completion time goals for incoming workloads [12]; architecture adaptation [13] requires user creation of a model for adapting a known workload; and learning based approaches to date have been limited to learning a single scheduling policy for a specific architecture [14]. Our research explores a new approach to scheduling within DEFs, an Emergent Scheduler, capable of autonomously selecting and composing the ideal architecture and policy for a given workload at runtime, drawing on a large pool of potential building blocks.

- Can we use behavioural composition (instead of policy adaptation) to form different optimal schedulers for different workloads / environments?
- Can we learn the optimal composition at runtime? How much performance gain do we get vs a fixed architecture?

We present initial results which demonstrate the performance of different scheduling strategies when being subjected to workloads of varying granularity, indicating different scheduling policies are better suited to different workloads. The results represent an interesting runtime search space for learning the ideal composition for a given workload. However, the learning process in this domain is uniquely challenging in that workloads can be highly divergent with few shared characteristics, are often non-repeating, and their arrival can be sporadic. This is a key area of future work, in addition to considering storage locations of data and more diverse scheduling architectures.

REFERENCES

[1] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[2] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan *et al.*, "Ray: A distributed framework for emerging {AI} applications," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 561–577.

[3] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *ArXiv*, vol. abs/1802.05799, 2018.

[4] 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

[5] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, 2015.

[6] M. Armbrust, T. Das, J. Torres, B. Yavuz, S. Zhu, R. Xin, A. Ghodsi, I. Stoica, and M. Zaharia, "Structured streaming: A declarative api for real-time applications in apache spark," in *Proceedings of the 2018 International Conference on Management of Data*, ser. SIGMOD '18. New York, NY, USA: ACM, 2018, pp. 601–613. [Online]. Available: http://doi.acm.org/10.1145/3183713.3190664

[7] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy, "Storm@twitter," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 147–156. [Online]. Available: http://doi.acm.org/10.1145/2588555.2595641

[8] I. Gog, M. Schwarzkopf, A. Gleave, R. N. Watson, and S. Hand, "Firmament: Fast, centralized cluster scheduling at scale," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 99–115.

[9] E. Boutin, J. Ekanayake, W. Lin, B. Shi, J. Zhou, Z. Qian, M. Wu, and L. Zhou, "Apollo: scalable and coordinated scheduling for cloud-scale computing," in *Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2014, pp. 285–300. [Online]. Available: https://dl.acm.org/citation.cfm?id=2685071

[10] K. Ousterhout, P. Wendell, and M. Zaharia, "Sparrow: Distributed, Low Latency Scheduling." [Online]. Available: http://dx.doi.org/10.1145/2517349.2522716

[11] P. Delgado, F. Dinu, A.-M. Kermarrec, and W. Zwaenepoel, "Hawk: Hybrid Datacenter Scheduling," in *2015 {USENIX} Annual Technical Conference ({USENIX} {ATC} 15)*. Santa Clara, CA: USENIX Association, 2015, pp. 499–510. [Online]. Available: https://www.usenix.org/conference/atc15/technical-session/presentation/delgado

[12] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguadé, "Resource-aware adaptive scheduling for mapreduce clusters," in *Middleware 2011*, F. Kon and A.-M. Kermarrec, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 187–207.

[13] Y. Xia, R. Ren, H. Cai, A. V. Vasilakos, and Z. Lv, "Daphne: A flexible and hybrid scheduling framework in multi-tenant clusters," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 330–343, March 2018.

[14] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," 2018.