Learning to Optimize Plan Execution in Information Agents

Craig A. Knoblock

University of Southern California, Information Sciences Institute,
4676 Admiralty Way, Marina del Rey, CA 90292
knoblock@isi.edu

Overview

We can build software agents to perform a wide variety of useful information gathering and monitoring tasks on the Web [1]. For example, in the travel domain, we can construct agents to notify you of flight delays in real time, monitor for schedule and price changes, and even send a fax to a hotel if your flight is delayed to ensure that your hotel room will not be given away [2,3].

To perform each of these tasks, an agent is given a plan and its needs to be able to efficiently execute this plan. In the Web environment, sources can be quite slow and the latencies of the sources are also unpredictable since they can be caused by heavy loads on both servers and networks. Since the primary bottleneck of most agent plans on the web is retrieving data from online sources, an agent should execute information requests as early as possible. To address these issues, we have developed a streaming dataflow language and executor, called Theseus [4], which is optimized for the Web environment in three ways. First, since the executor is based on a dataflow paradigm, actions are executed as soon as the data becomes available. Second, Theseus performs the actions in a plan in separate threads, so they can be run asynchronously and in parallel. Third, the system streams the output from one action to the next so that sequential operations can be executed in parallel.

Theseus is similar to network query engines, such as Telegraph [5] or Tukwila [6], in that they are also streaming dataflow execution systems. However, the network query engines focus on the efficient execution of XML queries, while Theseus provides an expressive language for expressing information gathering and monitoring plans. The Theseus language supports capabilities that go beyond network query engines in that it supports recursion, notification operations, and writing and reading from databases to support monitoring tasks.

We developed an approach to increase the potential parallelism in a streaming dataflow execution system. This optimization technique, called speculative execution [7,8], predicts the results of an operation based on data and patterns that it has seen in the past. The predicted results can then be used to speculate about the operations that will need to be performed later in the plan. The system decides where to speculate by analyzing a plan and determining the critical paths. On these paths it then inserts a "speculate" operation, which uses input to earlier operations to predict the input to later operations. The system also inserts a "confirm" operation, which ensures that the final result is correct regardless of whether the prediction is correct. This approach to optimizing streaming dataflow plans can achieve arbitrary speedups by speculating on the speculations. If the system is able to make accurate predictions, the executor could speculate on all of the input, execute the entire plan in parallel, and then confirm all of the results.

The effectiveness of the speculation technique depends on making accurate predictions. We have developed a learning system that combines caching, classification, and transduction to learn value predictors [9]. The system uses transducer learning to discover patterns in Web navigation paths, decision tree learning to make predictions on similar inputs, and caching when the first two learning methods are not applicable. Our experiments on a number of real-world examples show that learning the value predictors for speculative execution can provide significant performance improvements over the same plans without any learning.

References

1. Knoblock, C.A.: Deploying information agents on the web. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003), Acapulco, Mexico (2003)
2. Ambite, J.L., Barish, G., Knoblock, C.A., Muslea, M., Oh, J., Minton, S.: Getting from here to there: Interactive planning and agent execution for optimizing travel. In: Proceedings of the Fourteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-2002), AAAI Press, Menlo Park, CA (2002)
3. Chalupsky, H., Gil, Y., Knoblock, C.A., Lerman, K., Oh, J., Pynadath, D.V., Russ, T.A., Tambe, M.: Electric elves: Applying agent technology to support human organizations. In: Proceedings of the Conference on Innovative Applications of Artificial Intelligence. (2001)
4. Barish, G., Knoblock, C.A.: An expressive and efficient language for software agents. Journal of Artificial Intelligence Research (2005)
5. Hellerstein, J.M., Franklin, M.J., Chandrasekaran, S., Deshpande, A., Hildrum, K., Madden, S., Raman, V., Shah, M.A.: Adaptive query processing: Technology in evolution. IEEE Data Engineering Bulletin **23** (2000)
6. Ives, Z.G., Halevy, A.Y., Weld, D.S.: An XML query engine for network-bound data. VLDB Journal **11** (2002)
7. Barish, G., Knoblock, C.A.: Speculative execution for information gathering plans. In: Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002), AAAI Press, Menlo Park, CA (2002)
8. Barish, G., Knoblock, C.A.: Learning value predictors for the speculative execution of information gathering plans. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003), Acapulco, Mexico (2003)
9. Barish, G.: Speculative Plan Execution for Information Agents. PhD thesis, Department of Computer Science, University of Southern California (2003)