If All Else Fails, Read the Instructions! A Perspective on "GPTuner: An LLM-Based Database Tuning System".

Immanuel Trummer

The performance of a database management system depends on various tuning parameters. From the amount of memory allocated to certain operators over the degree of parallelism used in specific scenarios up to logging and recovery behavior, almost every aspect of data processing can be influenced by setting parameters accordingly. Those settings can have a significant impact on the performance of database systems. Default settings tend to work badly for any given scenario. That motivates the question: how can we find parameter settings to optimize a specific performance metric (e.g., run time or throughput) for a specific workload?

Following the well-known saying "If all else fails, read the instructions!", a human database administrator would probably start by reading the database manual. It explains the semantics of various parameters and provides first hints on how to set them appropriately for specific workloads and hardware types. After reading the manual, the human administrator might start browsing the web for tutorials on database tuning, focusing primarily on tutorials that are relevant to the specific database system, hardware and software platform, and workload type of interest. After that, the administrator might still skim a few blog entries, or related discussions on web forums like Stack Overflow. After digesting all of that relevant information, the administrator might try out a few combinations of parameter settings, informed by the hints obtained from web text. For those few settings, the administrator would probably evaluate the performance on a representative workload sample and select the best alternatives.

Until quite recently, automated tools for database tuning [3, 2] would miss out on all of these valuable sources of information. The space of possible parameter settings is large, integrating hundreds of tuning parameters for mature database systems such as PostgreSQL and MySQL. Unlike other optimization problems in the database domain, no analytical models are available that link tuning choices to performance estimates. This means that finding optimal settings requires actually executing and evaluating parameter settings by running example workloads, e.g., in the context of a reinforcement learning framework [2] (one of the most popular approaches to database tuning). Without further guidance enabling tools to prune and prioritize the search space, this approach is extremely expensive as it requires many trial runs to find good configurations.

The advent of large language models (LLMs) opens up new possibilities for database tuning. With almost human-level text processing abilities, they unlock text documents as an additional source of information for database tuning. In other words, the saying "If all else fails, read the instructions!" now applies to automated tuning tools as well. GPTuner [1] exploits LLMs to extract useful hints for database tuning, e.g., a recommendation to set specific tuning parameters to specific values. Then, it explores a search space of reduced size, informed by the information extracted from text, to find good configurations quickly.

The GPTuner system dethrones DB-BERT [4], the system pioneering the use of LLMs for database tuning, by innovating along multiple axes. First, it exploits the capabilities of modern LLMs, including the likes of GPT-4, to gain more information for database tuning, compared to prior work in this space. For instance, it exploits LLMs to uncover inconsistencies between tuning recommendations made by different sources. In addition, it takes into account the reliability of different sources when aggregating recommendations. Second, GPTuner innovates in the optimization phase, exploring a search space shaped by mined tuning hints via trial runs. Different from prior work in that space, notably DB-BERT, GPTuner replaces reinforcement learning with a Bayesian Optimization framework. To make this work well, GPTuner exploits LLMs again to reduce the number of considered tuning parameters and the value ranges considered for each of them. The resulting system achieves new state-of-the-art results for LLM-supported database tuning.

1. REFERENCES

[1] J. Lao, Y. Wang, Y. Li, J. Wang, Y. Zhang, Z. Cheng, W. Chen, M. Tang, and J. Wang. GPTuner: A Manual-Reading Database Tuning System via GPT-Guided Bayesian Optimization. *PVLDB*, 17(8):1939–1952, 2024.

[2] G. Li, X. Zhou, S. Li, and B. Gao. QTune: A QueryAware database tuning system with deep reinforcement learning. *PVLDB*, 12(12):2118–2130, 2018.

[3] A. Pavlo, G. Angulo, J. Arulraj, H. Lin, J. Lin, L. Ma, P. Menon, T. C. Mowry, M. Perron, I. Quah, S. Santurkar, A. Tomasic, S. Toor, D. V. Aken, Z. Wang, Y. Wu, R. Xian, and T. Zhang. Self-driving database management systems. In *CIDR*, pages 1–6, 2017.

[4] I. Trummer. DB-BERT: a Database Tuning Tool that "Reads the Manual". In *SIGMOD*, pages 190–203, 2022.