COMP90050 GROUP 16

Adaptive Query Processing

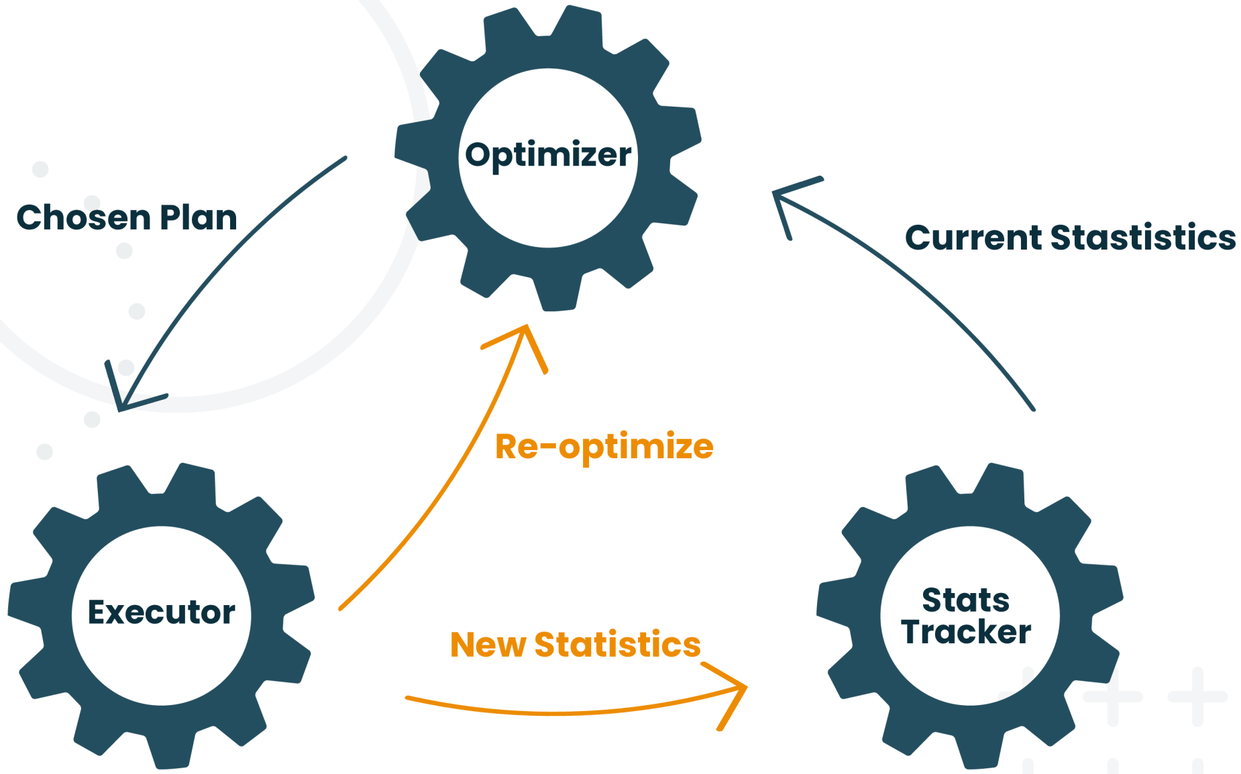
Liumingzi Zhu 727740
Xiaolu Zhang 886161
Luyao Chen 1266572
Muhan Guan 1407870

Table of Contents

Introduction

Traditional Query Processing

Adaptive Query Processing



Why AQP

01 Unreliable cardinality estimation

02 Changing data and system resources

03 Error propagation in complex queries

04 Queries in online environment

Related Works

01 Plan-based

02 Routing-based

03 Continuous Query-based

04 Reinforcement Learning-based

Plan-based

Parametric Query Optimization

- Precompute multiple plans for different situations and choose the best plan at run-time
- Examples
  - PQO (1994)
  - PPQO (2008)

Re-Optimization

- Use the optimizer to generate a better plan during query execution
- Examples
  - Re-Opt (1998)
  - POP (2024)
  - QuerySplit (2023)

Competition

- Run multiple plans in parallel and selects the most promising one
- Examples
  - DEC Rdb (1996)

Plan-based

	 Re-Optimization Frequency	 Performance Overhead	 Scalability
Competition	Low: one decision per table	High overhead: run multiple plans in parallel	Low scalability
Re-Optimization	**Re-Opt**: Low **POP**: High **QuerySplit**: Proactive	**Re-Opt**: Low **POP**: High **QuerySplit**: Moderate	**Re-Opt**: Low **POP**: Low **QuerySplit**: High
Parametric Query Optimization	**PQO**: Depends on condition change **PPQO**: Lower than PQO	**PQO**: Low during execution **PPQO**: Lower than PQO	**PQO**: Low due to high start-up costs **PPQO**: Moderate due to plan reuse

Routing-based

Eddy

(+) Dynamic operator ordering

(-) Sub-optimal routing decisions under **constraints on routing history**

SteMs

(+) Improved join performance
(+) Independence on intermediate state

(-) Constrained plan choices
(-) Re-computation of intermediate tuples

STAIRs

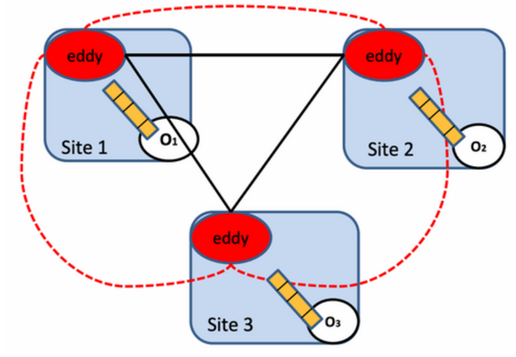
(+) Flexible storage and reuse of intermediate results

Routing-based

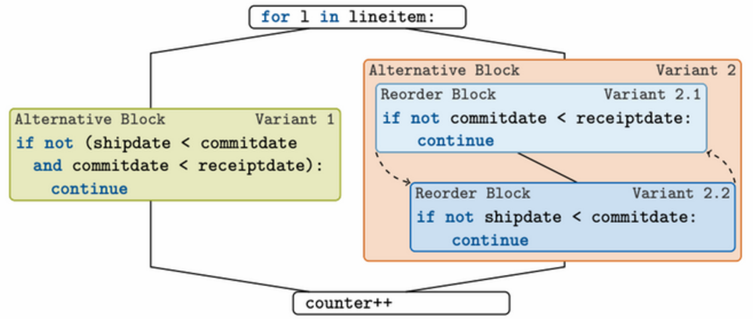
	 Execution Time	 State Handling	 Adaptivity
Eddy	Fast	Accumulates state in operators	Moderate
SteMs	Slower	No storage of intermediate results	Moderate
STAIRs	Faster for natural horizontal partitioning data	Storing long-living intermediate tuples	Higher, allows correction of earlier routing decisions

Routing-based

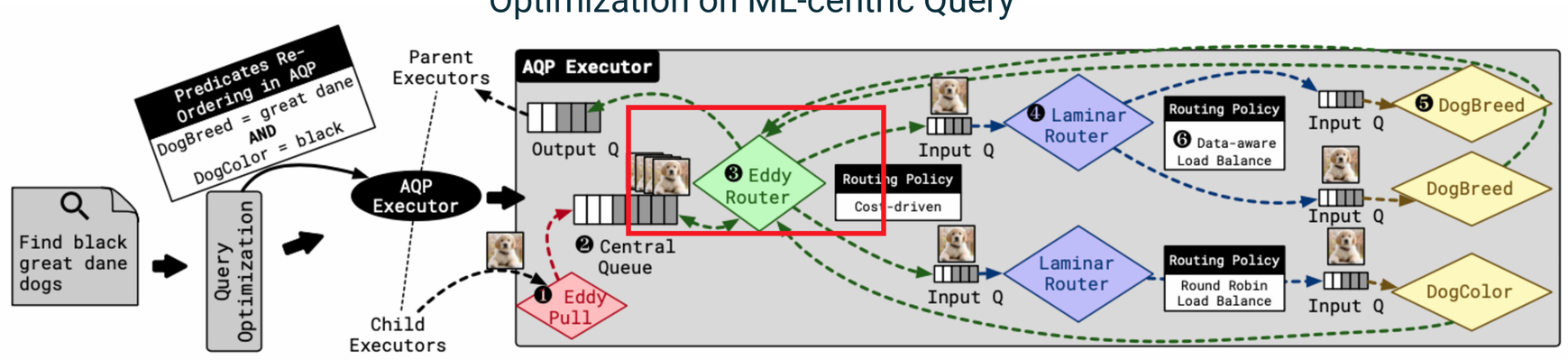
Eddy Architecture under Distributed Setting



Compiling Database Systems



Optimization on ML-centric Query



Continuous Query-based

Pure CQ-based

- Use Stream data characteristics
- Dynamically choose optimal plan
- Query sharing

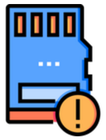
- Examples
  - **NiagaraCQ** (2000)
  - CAPE (2004)
  - StreaMon(2004)

Eddy + CQ-based

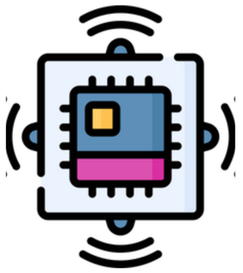
- Eddy based dynamic routing
- UC Berkeley’s Telegraph Projects

- Examples
  - **CACQ** (2002)
  - Psoup (2003)
  - TelegraphCQ(2003)

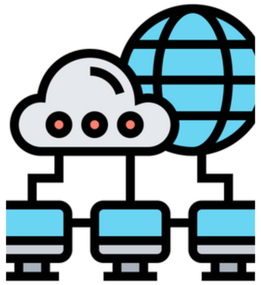
Continuous Query-based

	 Join Optimization	 Computational Cost	 Performance Evaluation
NiagaraCQ	Use **join signatures** to share join execution across multiple queries	(+) **Cache** group query plans and recently accessed files	(+) Works well with **overlapped queries** (-) Drops in **multiple complex queries**, especially in UDF scenarios
CACQ	Uses **SteMs** to enable pipelined multiway joins	(+) Use **grouped filter** to reduce computation	Works well with both scenarios

Continuous Query-based

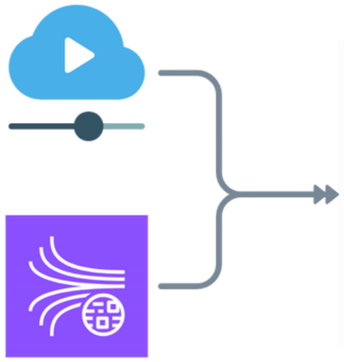


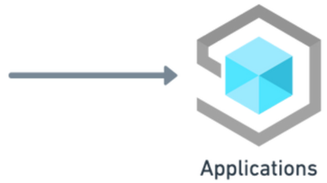
Sensor networks



Network monitors

Stock price alerts





Nowadays

Reinforcement Learning-based

2008	2018	2021	2021	2022
RL with Eddies	Cuttlefish	RouLette	**SkinnerDB**	**SkinnerMT**
			(+) Minimize execution overhead	(++) Address sequential join processing limitation
			(+) Converge to optimal join order	(+) Use multi-threading to improve scalability

Cross Category Comparison

Plan-based & CQ-based	vs	Routing-based & RL-based
Prediction and algorithm of statistics	Information collection	Information gathered during exploration
Using cost modeller for estimation	Cost estimation	Using greedy approach for cost estimation
Need to determine and choose re-optimization plans	Plan choosing	Re-optimization of plan happens automatically

Conclusion & Future Direction

Scalability

- Trade-off between granularity and runtime overhead
- Support distributed systems

Standardized Benchmark

- Build standardized benchmark for system comparison

Security and Reliability

- Support disaster recovery and fault tolerance

Evolution of Machine Learning and AI

- Incorporate ML and AI algorithms to develop intelligent query processing systems

THANKS

FOR YOUR ATTENTION

Reference

[1] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system," in Proceedings of the 1979 ACM SIGMOD international conference on Management of data, pp. 23–34, 1979.

[2] M. Stillger, G. M. Lohman, V. Markl, and M. Kandil, "Leo-db2's learning optimizer," in VLDB, vol. 1, pp. 19–28, 2001.

[3] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, "Query processing, resource management, and approximation in a data stream management system," in CIDR 2003, Stanford InfoLab, 2002.

[4] Y. E. Ioannidis and S. Christodoulakis, "On the propagation of errors in the size of join results," in Proceedings of the 1991 ACM SIGMOD International Conference on Management of data, pp. 268–277, 1991.

[5] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas, "Interactive data analysis: The control project," Computer, vol. 32, no. 8, pp. 51–59, 1999.

[6] Z. G. Ives, A. Y. Levy, D. S. Weld, D. Florescu, and M. Friedman, "Adaptive query processing for internet applications," 2000.

[7] S. Babu and P. Bizarro, "Adaptive query processing in the looking glass," in Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR), Jan. 2005.

[8] T. Schmidt, P. Fent, and T. Neumann, "Efficiently compiling dynamic code for adaptive query processing.," in ADMS@ VLDB, pp. 11–22, 2022.

[9] N. Kabra and D. J. DeWitt, "Efficient mid-query re-optimization of sub-optimal query execution plans," in Proceedings of the 1998 ACM SIGMOD international conference on Management of data, pp. 106–117, 1998.

[10] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh, and M. Cilimdzic, "Robust query processing through progressive optimization," in Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 659–670, 2004.

[11] J. Zhao, H. Zhang, and Y. Gao, "Efficient query re-optimization with judicious subquery selections," Proceedings of the ACM on Management of Data, vol. 1, no. 2, pp. 1–26, 2023.

[12] R. L. Cole and G. Graefe, "Optimization of dynamic query evaluation plans," in Proceedings of the 1994 ACM SIGMOD international conference on Management of data, pp. 150–160, 1994.

[13] P. Bizarro, N. Bruno, and D. J. DeWitt, "Progressive parametric query optimization," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 4, pp. 582–594, 2008.

[14] G. Antoshenkov and M. Ziauddin, "Query processing and optimization in oracle rdb," The VLDB Journal, vol. 5, pp. 229–237, 1996.

[15] A. Deshpande, Z. Ives, V. Raman, et al., "Adaptive query processing," Foundations and Trends® in Databases, vol. 1, no. 1, pp. 1–140, 2007.

[16] R. Avnur and J. M. Hellerstein, "Eddies: Continuously adaptive query processing," in Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 261–272, 2000.

[17] V. Raman, A. Deshpande, and J. Hellerstein, "Using state modules for adaptive query processing," in Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405), pp. 353–364, 2003.

[18] A. Deshpande and J. M. Hellerstein, "Lifting the burden of history from adaptive query processing," in Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30, VLDB '04, p. 948–959, VLDB Endowment, 2004.

[19] B. R˘aducanu, P. Boncz, and M. Zukowski, "Micro adaptivity in vectorwise," in Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD '13, (New York, NY, USA), p. 1231–1242, Association for Computing Machinery, 2013.

[20] D. Terry, D. Goldberg, D. Nichols, and B. Oki, "Continuous queries over append-only databases," Acm Sigmod Record, vol. 21, no. 2, pp. 321–330, 1992.

[21] S. Madden, M. Shah, J. M. Hellerstein, and V. Raman, "Continuously adaptive continuous queries over streams," in Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pp. 49–60, 2002.

[22] S. Chandrasekaran and M. J. Franklin, "Psoup: a system for streaming queries over streaming data," The VLDB Journal, vol. 12, pp. 140–156, 2003.

[23] E. A. Rundensteiner, L. Ding, T. Sutherland, Y. Zhu, B. Pielech, and N. Mehta, "Cape: Continuous query engine with heterogeneous-grained adaptivity," in Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, pp. 1353–1356, 2004.

[24] M. A. Shah, J. M. Hellerstein, S. Chandrasekaran, and M. J. Franklin, "Flux: An adaptive partitioning operator for continuous query systems," in Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405), pp. 25–36, IEEE, 2003.

[25] S. Krishnamurthy, S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Madden, F. Reiss, and M. A. Shah, "Telegraphcq: An architectural status report," IEEE Data Eng. Bull., vol. 26, no. 1, pp. 11–18, 2003.

[26] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang, "Niagaracq: A scalable continuous query system for internet databases," in Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 379–390, 2000.

[27] S. Babu and J. Widom, "Streamon: an adaptive engine for stream query processing," in Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 931–932, 2004.

[28] K. Tzoumas, T. Sellis, and C. S. Jensen, "A reinforcement learning approach for adaptive query processing," History, pp. 1–25, 2008.

[29] T. Kaftan, M. Balazinska, A. Cheung, and J. Gehrke, "Cuttlefish: A lightweight primitive for adaptive query processing," arXiv preprint arXiv:1802.09180, 2018.

[30] I. Trummer, J. Wang, Z. Wei, D. Maram, S. Moseley, S. Jo, J. Antonakakis, and A. Rayabhari, "Skinnerdb: Regret-bounded query evaluation via reinforcement learning," ACM Trans. Database Syst., vol. 46, sep 2021.

[31] P. Sioulas and A. Ailamaki, "Scalable multi-query execution using reinforcement learning," in Proceedings of the 2021 International Conference on Management of Data, pp. 1651–1663, 2021.

[32] Z. Wei and I. Trummer, "Skinnermt: Parallelizing for efficiency and robustness in adaptive query processing on multicore platforms," Proc. VLDB Endow., vol. 16, p. 905–917, dec 2022.

[33] G. T. Kakkar, J. Cao, A. Sengupta, J. Arulraj, and H. Kim, "Hydro: Adaptive query processing of ml queries," arXiv preprint arXiv:2403.14902, 2024.

[34] A. Gounaris, E. Tsamoura, and Y. Manolopoulos, "Adaptive query processing in distributed settings," Intelligent Systems Reference Library, vol. 36, 01 2013.