POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Adaptive Tracing: Problematic Area Localization

By Masoumeh Nourollahi

16 May 2022

Michel DAGENAIS, Research supervisor
Naser EZZATI-JIVAN, Research co-supervisor

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Introduction

Large scale tracing challenges

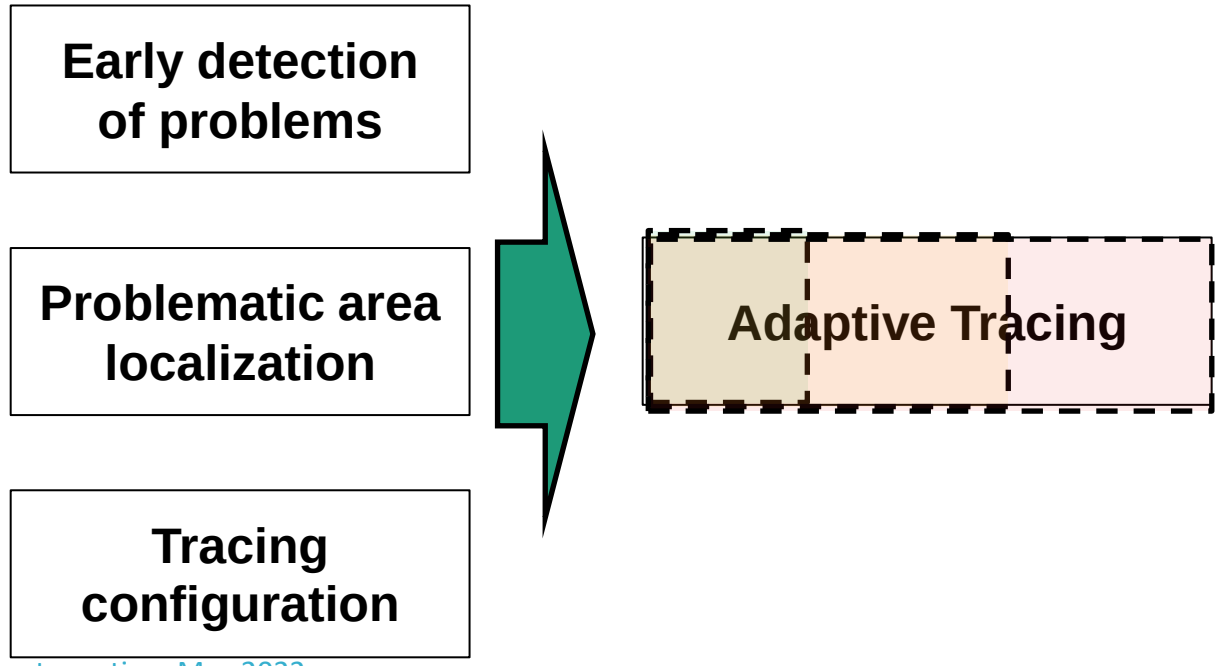
- Huge number of requests results in enormous traces
- Puts overhead in trace collection, storage, and analysis
- Not much intelligence in collecting traces

To improve tracing effectiveness,
tracing focus should adjust and adapt to collecting relevant events around the issues.

Research question

1. Can we increase the tracing effectiveness using trace adjustment methods at runtime, so that tracing is more focused on collecting events around the issues?

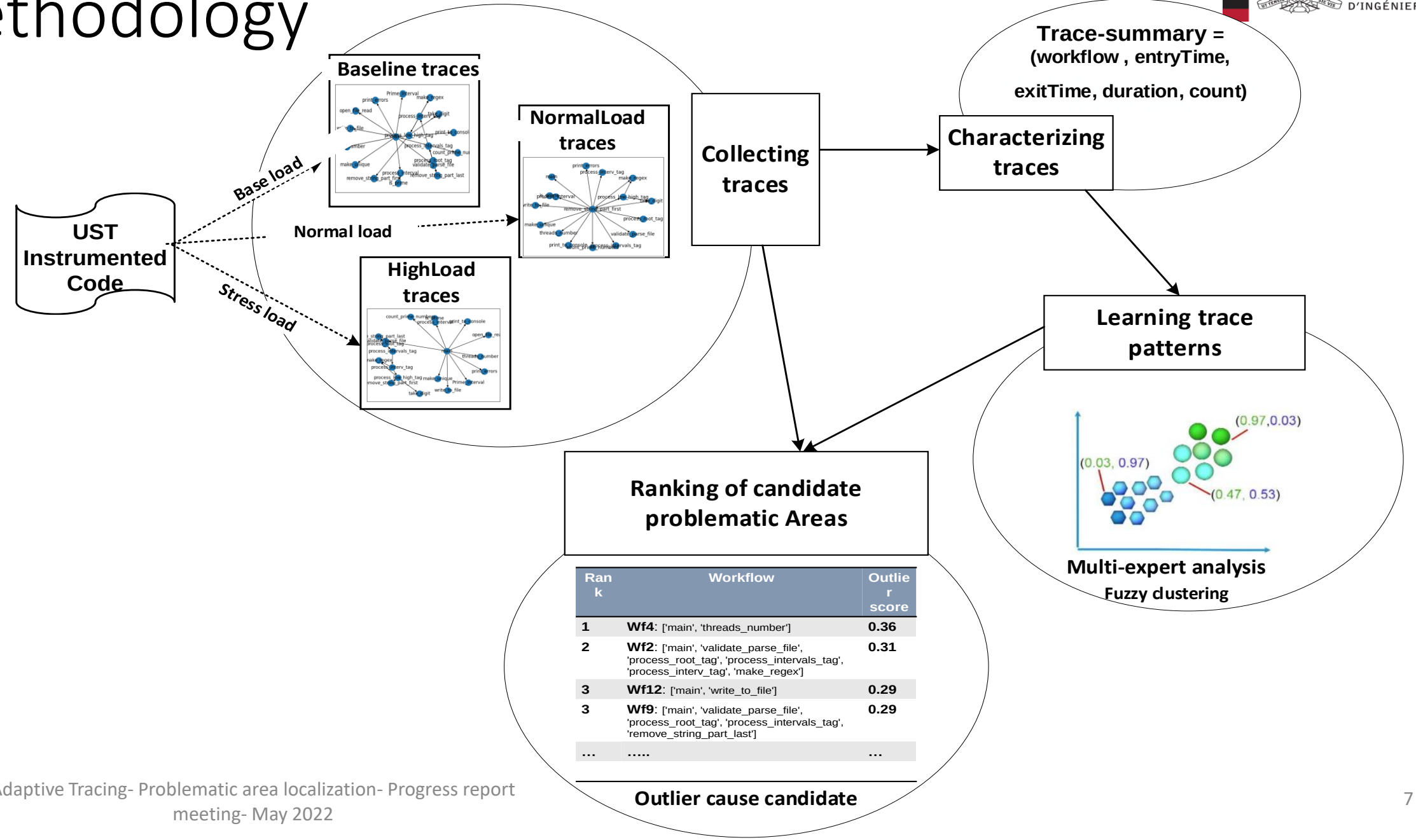
2. Can we identify the possible problematic areas by analyzing workload and resource metrics?



Similar workflows should perform similarly!

Preliminary results

Methodology



Rank	Workflow	Outlier score
1	**Wf4**: ['main', 'threads_number']	0.36
2	**Wf2**: ['main', 'validate_parse_file', 'process_root_tag', 'process_intervals_tag', 'process_interv_tag', 'make_regex']	0.31
3	**Wf12**: ['main', 'write_to_file']	0.29
3	**Wf9**: ['main', 'validate_parse_file', 'process_root_tag', 'process_intervals_tag', 'remove_string_part_last']	0.29
…	…..	…

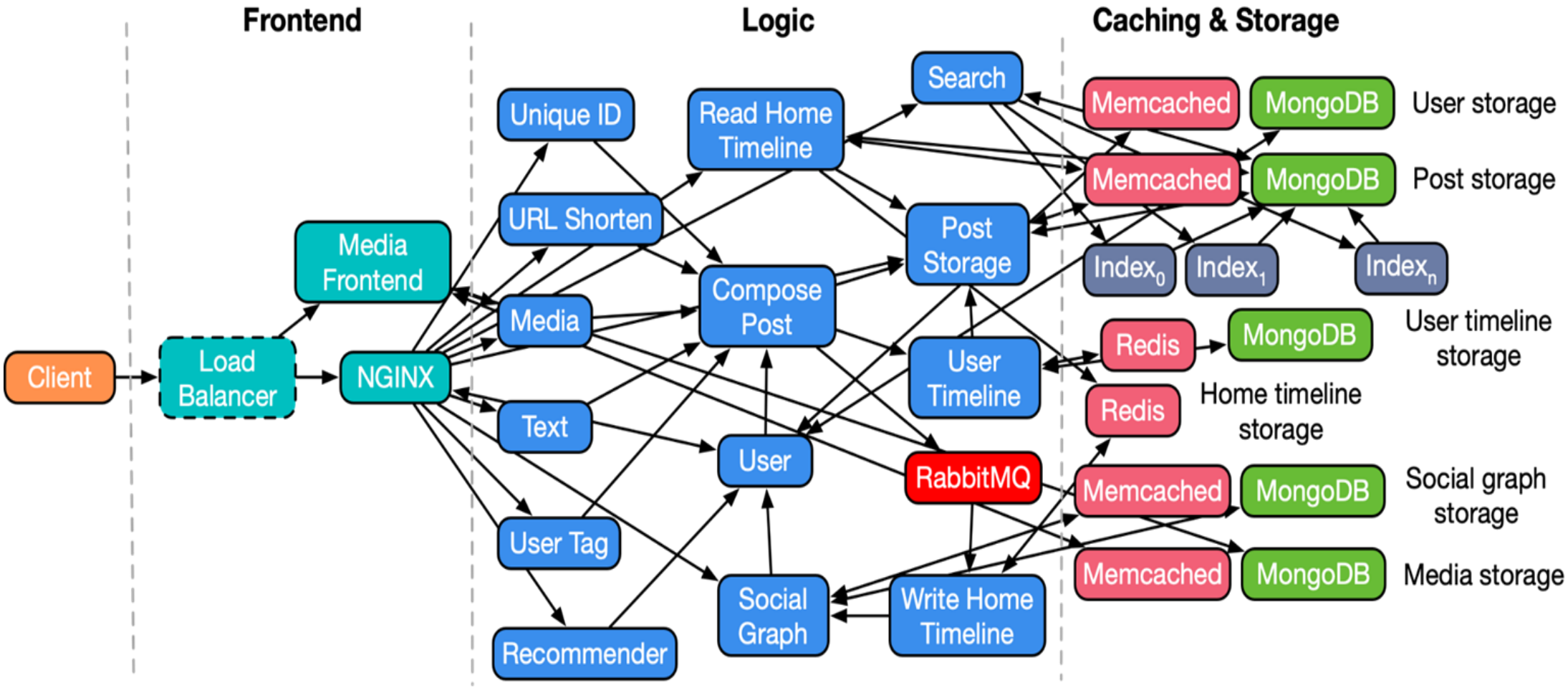
Analysis results

- Always slow workflows
  - Candidates for problematic areas. Should check code for the functions included in the workflows
- Slow workflows in high loads
  - Candidates for scalability issues. Should check in combination with resource usage
- Always freq. workflows
  - If performing well good candidates for trace sampling less frequently
- High load freq. workflow
  - Good candidates to check in-combination with workflows that are slow in high loads
- Less frequent workflows
  - If performing well good candidates to disable tracing

Demo

Evaluation setup

DeathStarBench (SocialNetwork)- test setup



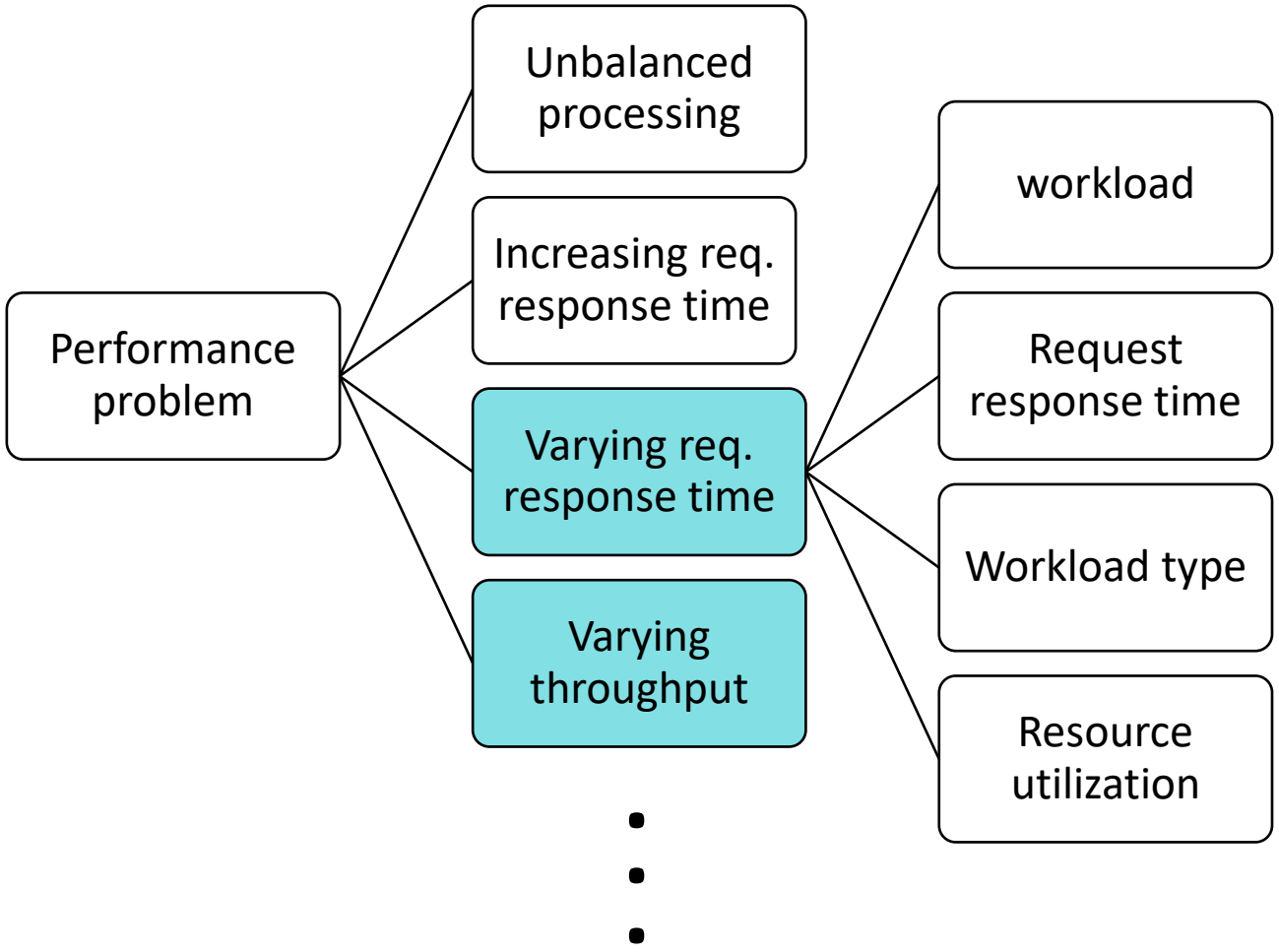
Characteristics of DeathstarBench- Social Network Service

- Supported actions in DeathStarBench
  - Create text post (optional media: image, video, shortened URL, user tag)
  - Read post
  - Read entire user timeline
  - Receive recommendations on which users to follow
  - Search database for user or post
  - Register/Login using user credentials
  - Follow/Unfollow user

Service	Total LoCs	Communication Protocol	Unique Microservices	Pre-language LoC breakdown
Social Network	**68061**	**RPC**	**36**	**34% C, 23% C++, 18% Java, 7% node.js, 6% Python, 5% Scala, 3% PHP, 2% Javascript, 2% Go**

Code Composition

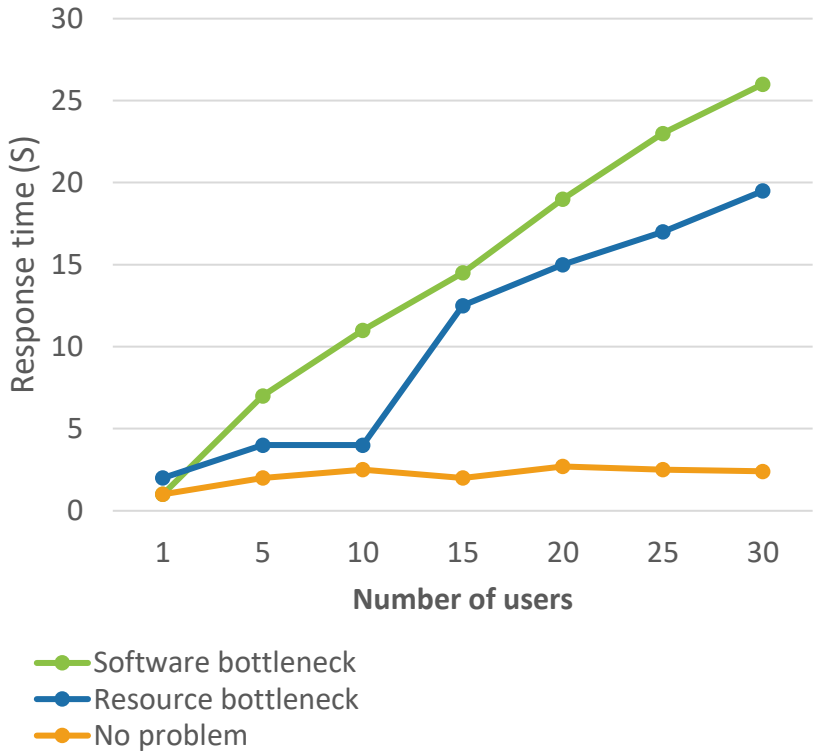
Problematic area localization

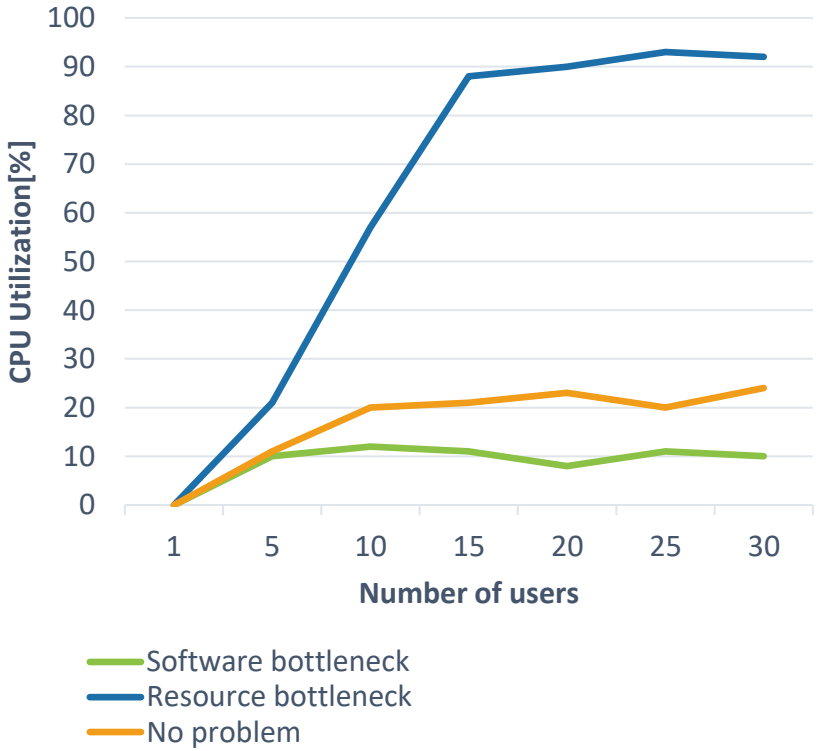


Performance problem:

- Scalability problem
  - Software bottleneck
    - Synchronization
    - External call bottleneck
  - Resource bottleneck
  - …..

Example: Software Bottleneck





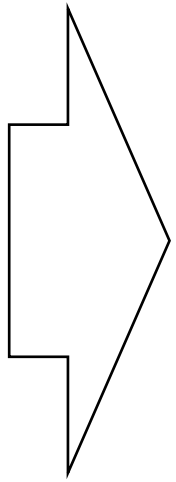
Checking software bottleneck detection

Software bottleneck possible causes:

- Synchronization points
- Database locks
- Pools
- ….

+

Related functions contain DB queries



Iinject DB Lock bottleneck in source-code

The way forward - update from the last meeting

- Automate the pipeline for the presented method
  - done
- Test and extend the method for more complex applications with longer execution paths
  - in-progress
- Investigate other methods to model frequency and duration metrics
  - working on fuzzy methods like fuzzy clustering
- Apply the same concept to other metrics like resource-related metrics modeling in combination with UST trace metrics
  - planned
- Improve performance of the code
  - in-progress

Github address for source code and test data:

https://github.com/mnourollahi/UST_adaptiveTracing

References

[1] Tânia Esteves, Francisco Neves, Rui Oliveira, and João Paulo. 2021. CAT: content-aware tracing and analysis for distributed systems. In Proceedings of the 22nd International Middleware Conference (Middleware '21). Association for Computing Machinery, New York, NY, USA, 223–235.

[2] S. Zhang, D. Liu, L. Zhou, Z. Ren, and Z. Wang, “Diagnostic framework for distributed application performance anomaly based on adaptive instrumentation,” in 2020 2nd International Conference on Computer Communication and the Internet (ICCCI). IEEE, 2020, pp. 164–169.

[3]E. Ates, L. Sturmann, M. Toslali, O. Krieger, R. Megginson, A. K. Coskun, and R. R. Sambasivan, “An automated, cross-layer instrumentation framework for diagnosing performance problems in distributed applications,” in Proceedings of the ACM Symposium on Cloud Computing, ser. SoCC ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 165–170.

[4] P. Las-Casas, J. Mace, D. Guedes, and R. Fonseca, “Weighted sampling of execution traces: Capturing more needles and less hay,” in Proceedings of the ACM Symposium on Cloud Computing, 2018, pp. 326–332.

[5] P. Las-Casas, G. Papakerashvili, V. Anand, and J. Mace, “Sifter: Scalable sampling for distributed traces, without feature engineering,” in Proceedings of the ACM Symposium on Cloud Computing, 2019, pp. 312–324.

[6] A. Bento, J. Correia, R. Filipe, F. Araujo, and J. Cardoso, “Automated analysis of distributed tracing: Challenges and research directions,” Journal of Grid Computing, vol. 19, no. 1, pp. 1–15, 2021.

[7] Q. Fournier, N. Ezzati-jivan, D. Aloise, and M. R. Dagenais, “Automatic cause detection of performance problems in web applications,” in 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2019, pp. 398–405.

[8] M. Dymshits, B. Myara, and D. Tolpin, “Process monitoring on sequences of system call count vectors,” in 2017 International Carnahan Conference on Security Technology (ICCST). IEEE, 2017, pp. 1–5.

[9] F. Doray and M. Dagenais, “Diagnosing performance variations by comparing multilevel execution traces,” IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 2, pp. 462–474, 2016.

[10] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, “Dapper, a large-scale distributed systems tracing infrastructure,” Google, Inc., Tech. Rep., 2010.

References

[11] J. Kaldor, J. Mace, M. Bejda, E. Gao, W. Kuropatwa, J. O'Neill, K. W. Ong, B. Schaller, P. Shan, B. Viscomi, V. Venkataraman, K. Veeraraghavan, and Y. J. Song, "Canopy: An end-to-end performance tracing and analysis system," in Proceedings of the 26th Symposium on Operating Systems Principles, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 34–50.

[12] Wert, Alexander, Jens Happe, and Lucia Happe. "Supporting swift reaction: Automatically uncovering performance problems by systematic experiments." 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013.

[13] M. Gebai et M. R. Dagenais, "Survey and analysis of kernel and userspace tracers on linux : Design, implementation, and overhead," ACM Comput. Surv., vol. 51, no. 2, mars 2018.

[14] S. Tjandra, "Performance model extraction using kernel event tracing," Thèse de doctorat, Carleton University, 2019.

[15] R. R. Sambasivan, A. X. Zheng, M. De Rosa, E. Krevat, S. Whitman, M. Stroucken, W. Wang, L. Xu et G. R. Ganger, "Diagnosing performance changes by comparing request flows." dans NSDI, vol. 5, 2011, p. 1–1.

[16] Du, Min, et al. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.

[17] V. Cortellessa et L. Traini, "Detecting latency degradation patterns in service-based systems," dans Proceedings of the ACM/SPEC International Conference on Performance Engineering, 2020, p. 161–172.

[18] F. Doray et M. Dagenais, "Diagnosing performance variations by comparing multi-level execution traces," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 2, p. 462–474, 2016.