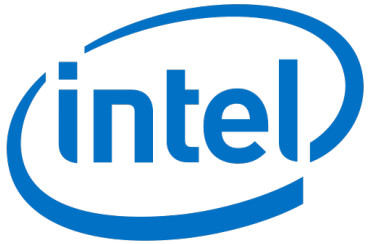
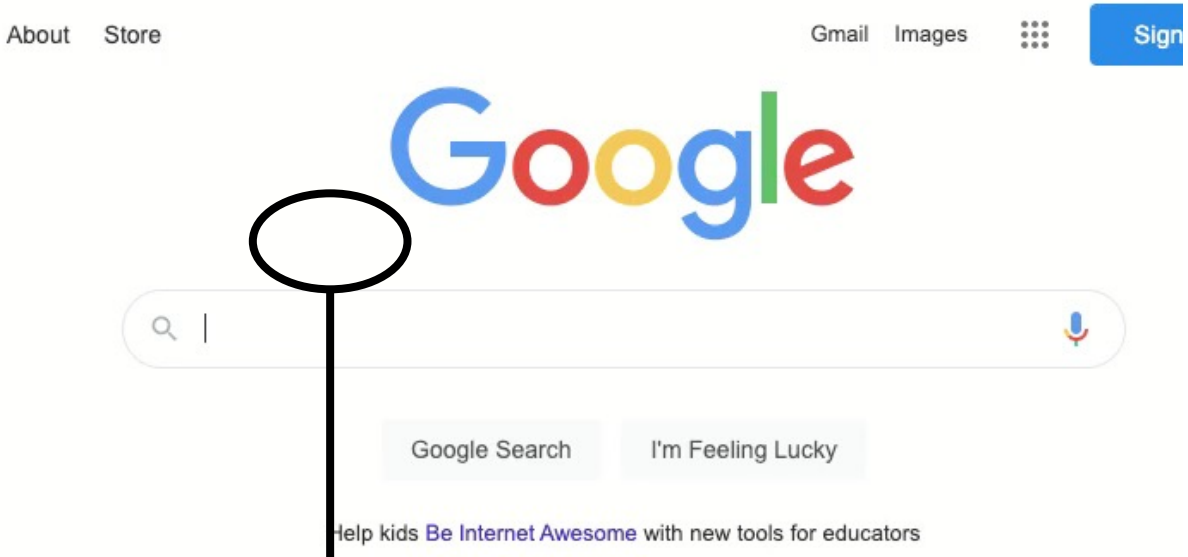
DMon: Efficient Detection and Correction of Data Locality Problems Using Selective Profiling

Tanvir Ahmed Khan, Ian Neal, Gilles Pokam, Barzan Mozafari, Baris Kasikci





~700 ms

Running that single search query requires 8 processor cores[1]!

Active usage of millions of processor cores to serve the planet!

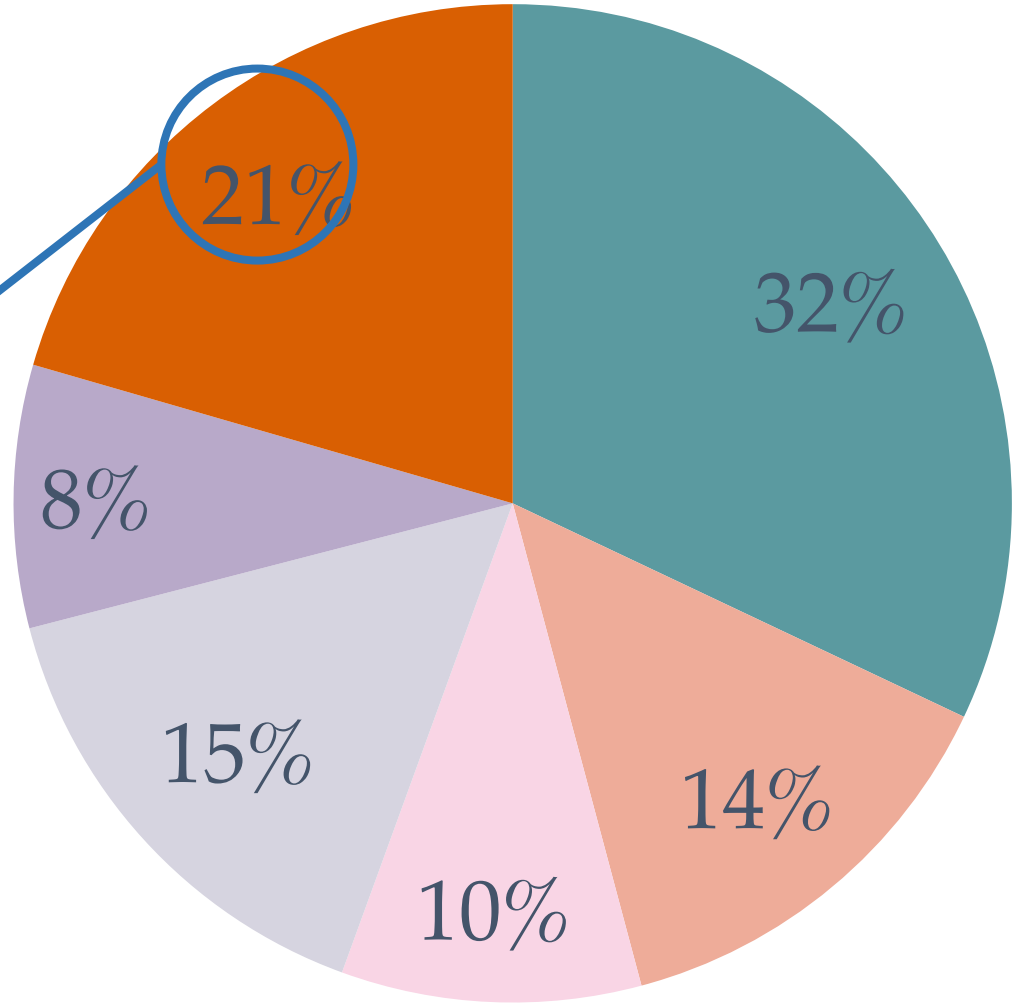
Millions of dollars in management and energy cost

+

Planet-scale carbon footprint

[1] Memory Hierarchy for Web Search, HPCA 2018

CPU Performance of Google Web Search[1]



Memory stalls due to poor data locality

- Doing useful work
- Fetching instructions
- Decoding instructions
- Bad speculation
- Execution units unavailable
- Memory stalls

[1] AsmDB: understanding and mitigating front-end stalls in warehouse-scale computers, ISCA 2019

Existing Techniques & Why They Fall Short?

Compiler Optimizations

- Automatically improve data locality via program transformation
- No run-time overhead
- Rely on static heuristics
- Can sometimes even hurt performance

Dynamic Profilers

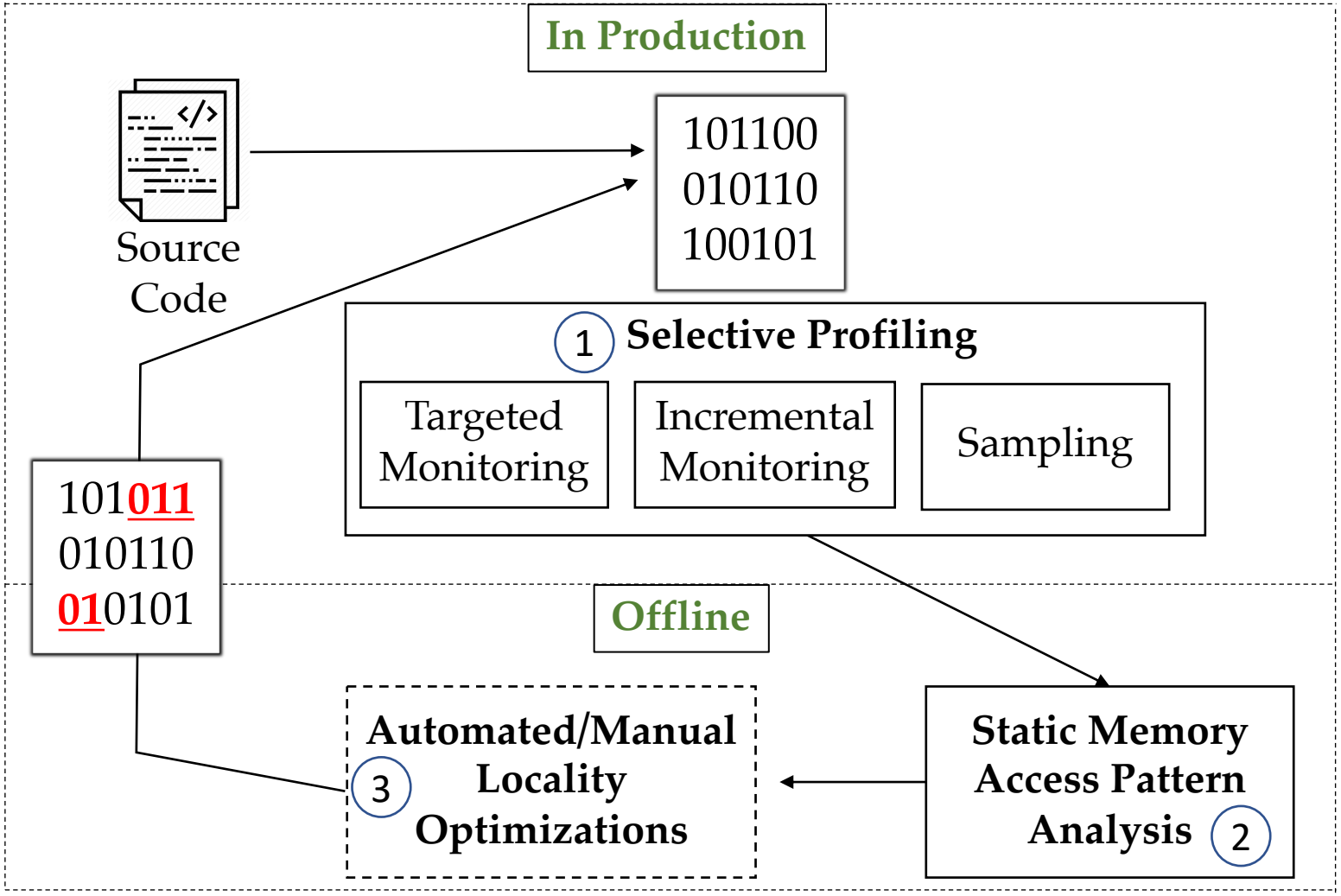
- Help developers identify and resolve poor data locality
- Accurate execution information
- Mostly manual repair
- High profiling overhead when used to detect data locality issues

DMon's Contributions

- Selective profiling to detect data locality problems accurately and efficiently
- Apply specific compiler optimizations based on profiling results
- Evaluation showing the efficiency of selective profiling and effectiveness of targeted optimizations
  - Negligible (less than 2%) overhead
  - 17% average speedup for popular benchmarks from PARSEC, SPLASH, NPB
  - 7% average speedup for PostgreSQL

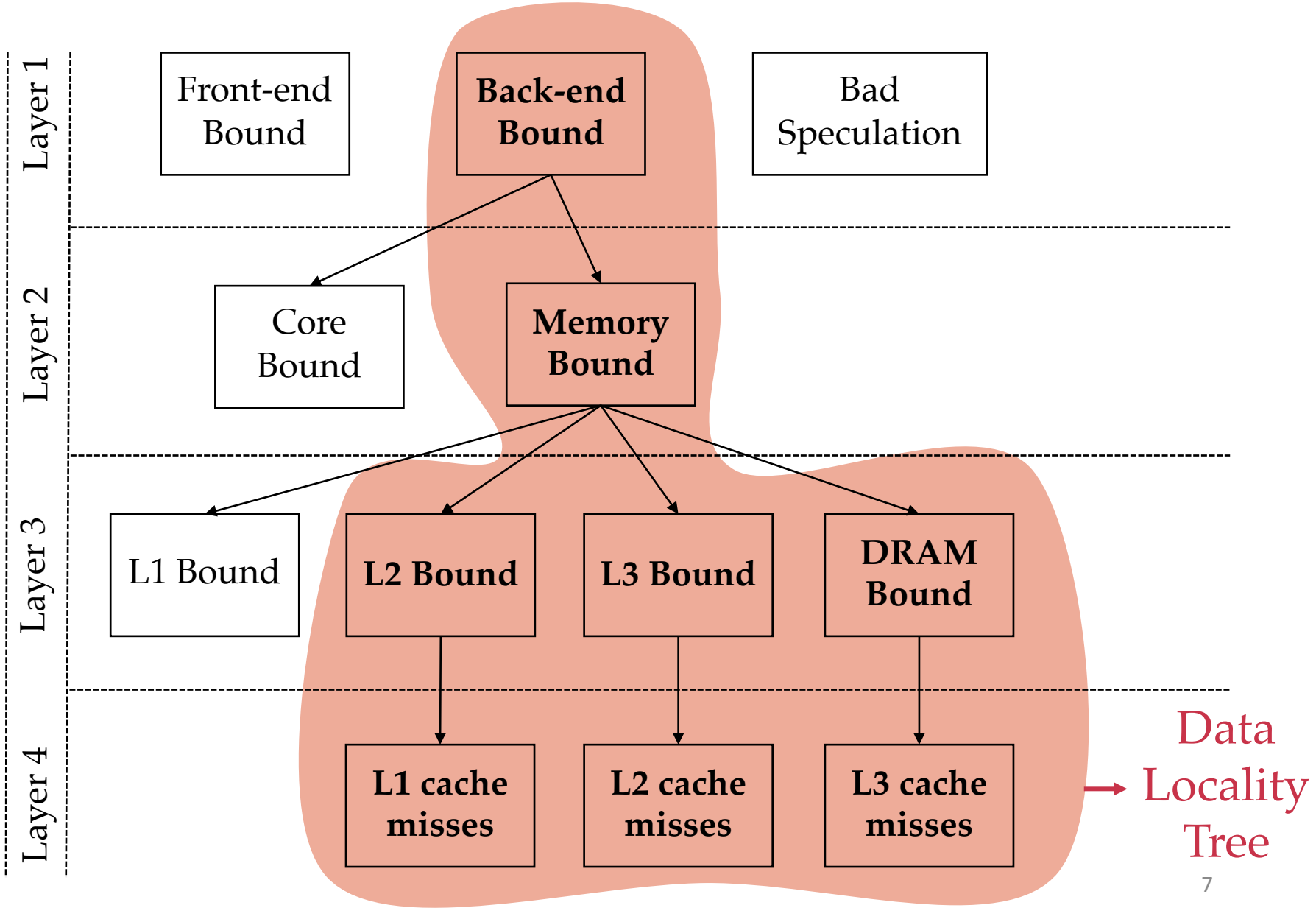
DMon's Design

- Continuous in-production monitoring to identify data locality problems
- In-house static analysis to identify memory access pattern
- In-house static transformations to optimize locality

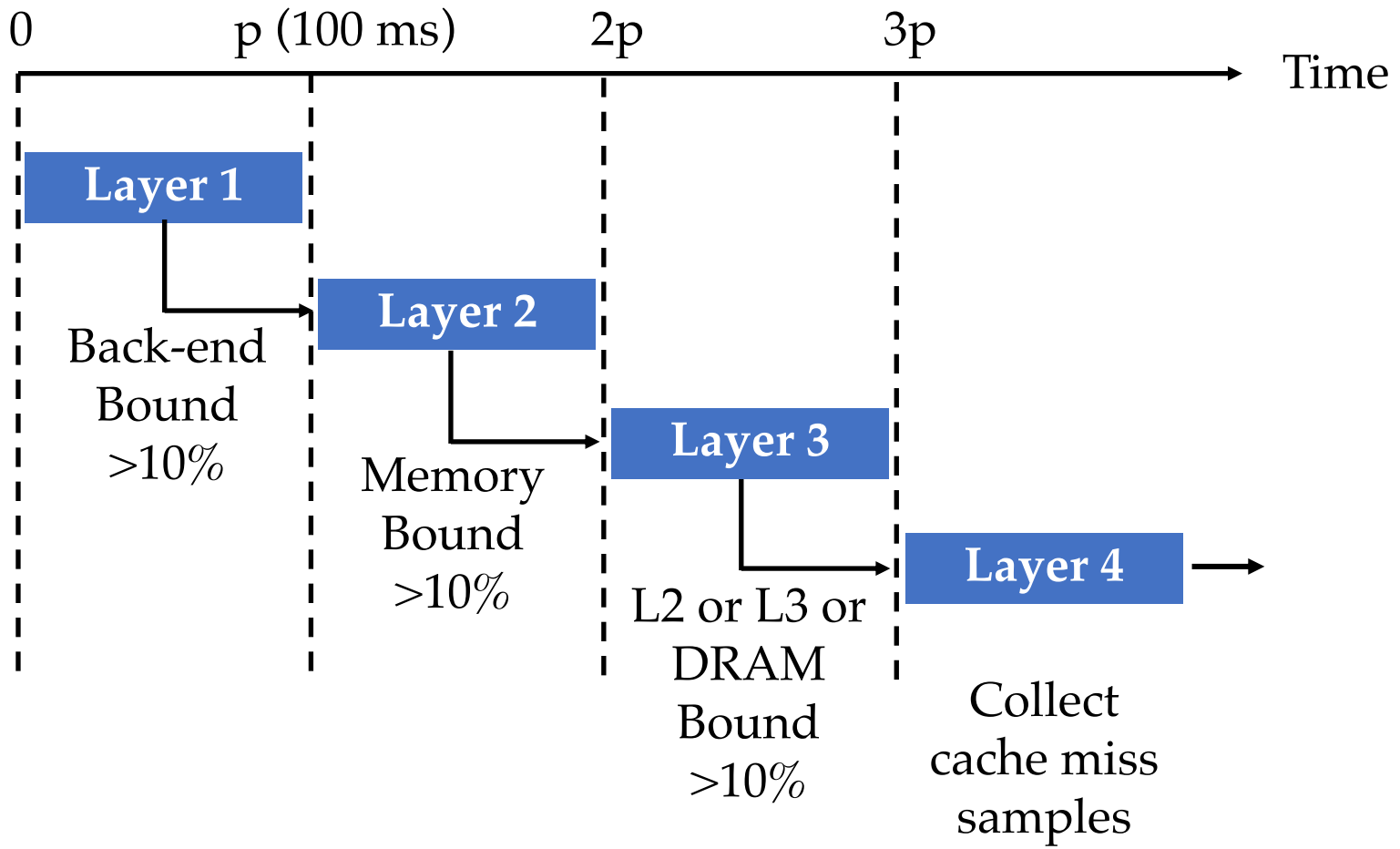


Targeted Monitoring

- Leverage the hierarchical Top-down approach from Intel
- Not all problems are related to data locality
- Only focus on a small subtree related to data locality

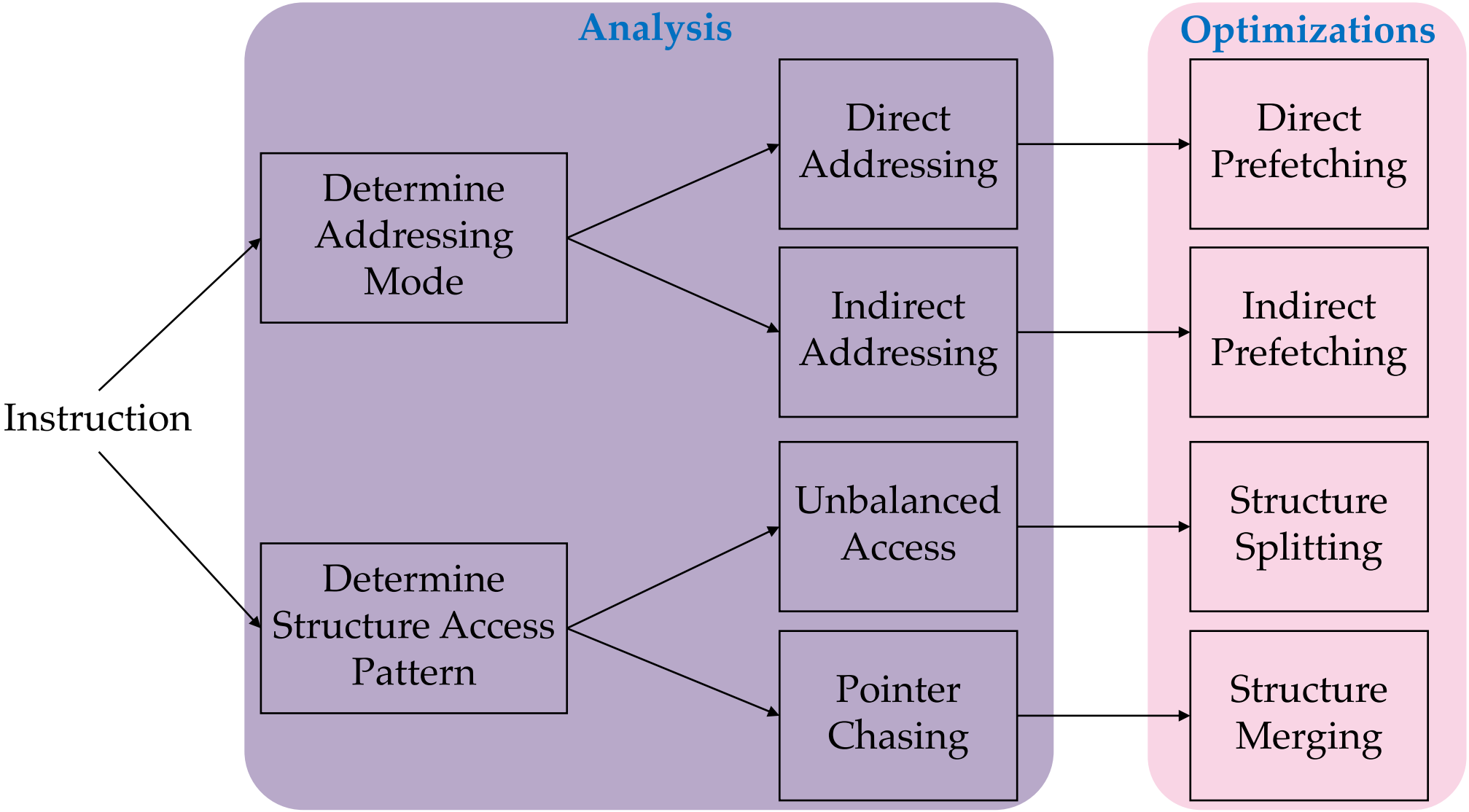


Incremental Monitoring



- Monitor the program execution in short time slices
- Incrementally enable more detailed profiling
- Can identify even different locality problems at various program phases

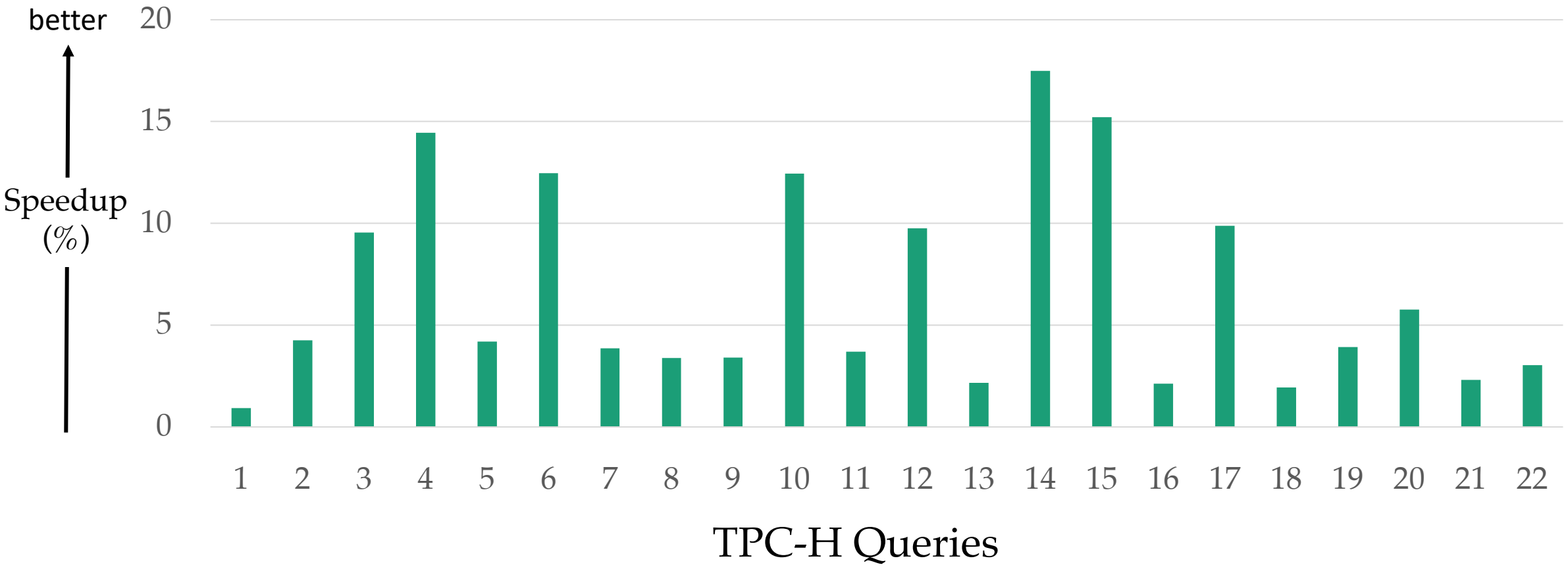
Offline Analysis and Transformations



Evaluation Summary

- Efficiency
  - On average 1.36% overhead
  - 9x lower overhead than state-of-the-art data locality profiler
- Effectiveness
  - Accurately detect data locality problems for benchmarks from PARSEC, SPLASH-2X, and NPB suites
  - On average 16.83% and up to 53.14% speedup
  - 20% more speedup than state-of-the-art profile-guided data locality optimizer
- Real-world case studies
  - PostgreSQL, Apache-spark page-rank, and others

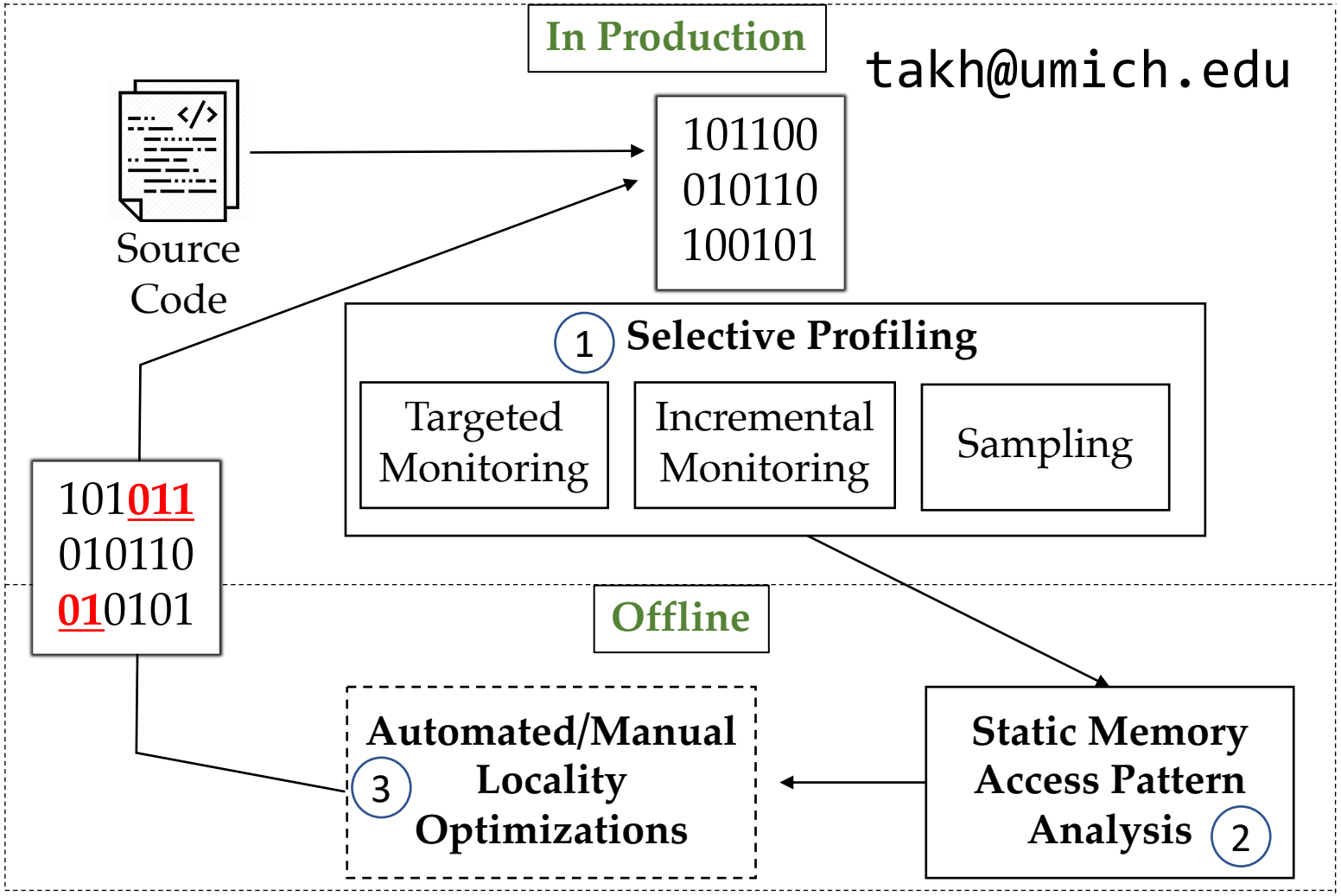
Performance Speedup on PostgreSQL



DMon speeds up PostgreSQL by 7% on average

DMon: Data Locality Optimizations via Selective Profiling

- Selective profiling to detect data locality problems accurately and efficiently
- Apply specific optimizations based on profiling results
- 17% speedup with negligible (less than 2%) overhead



github.com/efeslab/DMon-AE