**Sampling-Based AQP in Modern Analytical Engines**

Viktor Sanca and Anastasia Ailamaki

**Interactive analytics is an elusive goal**

- **Sampling in the critical path of execution**
  - CPU
  - Bottleneck shift
  - Throughput
  - Operator throughput mismatch
  - Detrimental and impractical? slower + approximate

**AQP + modern systems = faster analytics**

**Sampling in the critical path of execution**

- **Sampling in the critical path of execution**
  - CPU
  - Bottleneck shift
  - Throughput
  - Operator throughput mismatch
  - Detrimental and impractical? slower + approximate

**Common operations and access patterns**

- **Random number generation (RNG) throughput**
  - std::rand()
  - Optimized Mersenne
  - Lehmer Vectorized

**Exposing the bottlenecks: sampling inside an in-memory scale-up analytical engine**

**Setup:** dual socket Intel Xeon Gold 5118 (2x12 cores), 384GB RAM  
**Data:** SSB with 600M (SF100) and 6B (SF1000) tuples in fact table, 1 binary column has ~2.3/23GB

**Hardware- and algorithm-conscious sampling design**

Minimize the overheads in the query processing pipeline

Reduce the overhead to 1-2.5x over matching operators

**Sampling-based AQP is viable in modern engines**

- **Sampling in the critical path of execution**
  - CPU
  - Bottleneck shift
  - Throughput
  - Operator throughput mismatch
  - Detrimental and impractical? slower + approximate

**Interactive analytics is an elusive goal**

- **Sampling in the critical path of execution**
  - CPU
  - Bottleneck shift
  - Throughput
  - Operator throughput mismatch
  - Detrimental and impractical? slower + approximate

**AQP + modern systems = faster analytics**

**Sampling operator design**

- **Sampling operator design**
  - Filter
  - Bernoulli/Rejection
  - Aggregate
  - Reservoir
  - Group by Aggregate
  - Stratified

Seamless integration with the existing system

Just-in-time sampling triggered by query operators

Match the throughput of corresponding relational operators

**Goal: low overhead side-effect of query execution**

**Co-design the physical operators with algorithms**

Decouple state from data

State: frequent access

Data: infrequent updates

Better cache locality

**Sampling-based AQP is viable in modern engines**

- **Sampling in the critical path of execution**
  - CPU
  - Bottleneck shift
  - Throughput
  - Operator throughput mismatch
  - Detrimental and impractical? slower + approximate

**Interactive analytics is an elusive goal**

- **Sampling in the critical path of execution**
  - CPU
  - Bottleneck shift
  - Throughput
  - Operator throughput mismatch
  - Detrimental and impractical? slower + approximate

**AQP + modern systems = faster analytics**

**Sampling in the critical path of execution**

- **Sampling in the critical path of execution**
  - CPU
  - Bottleneck shift
  - Throughput
  - Operator throughput mismatch
  - Detrimental and impractical? slower + approximate

**Common operations and access patterns**

- **Random number generation (RNG) throughput**
  - std::rand()
  - Optimized Mersenne
  - Lehmer Vectorized

**Exposing the bottlenecks: sampling inside an in-memory scale-up analytical engine**

**Setup:** dual socket Intel Xeon Gold 5118 (2x12 cores), 384GB RAM  
**Data:** SSB with 600M (SF100) and 6B (SF1000) tuples in fact table, 1 binary column has ~2.3/23GB

**Hardware- and algorithm-conscious sampling design**

Minimize the overheads in the query processing pipeline

Reduce the overhead to 1-2.5x over matching operators

**Sampling-based AQP is viable in modern engines**