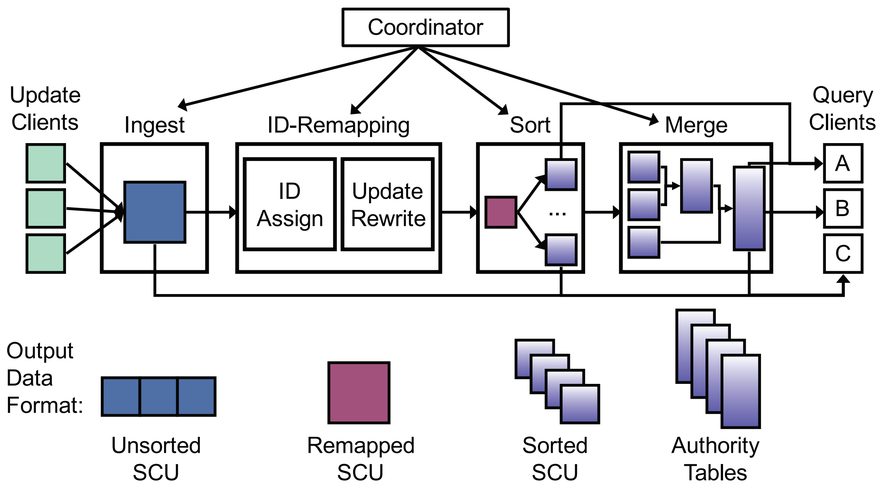
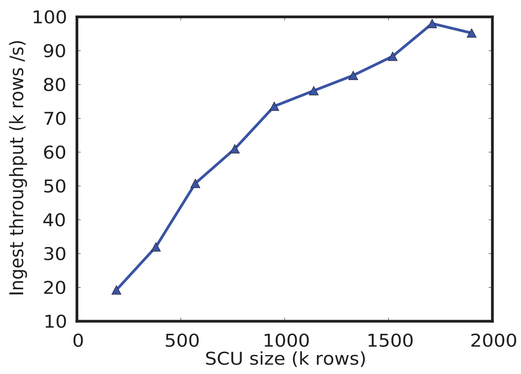
LAZYBASE: TRADING FRESHNESS FOR PERFORMANCE IN A SCALABLE DATABASE

Jim Cipar*, Greg Ganger*, Kim Keeton^, Brad Morrey^, Craig Soules^, Alistair Veitch^ (*CMU, ^HP)

LAZYBASE

- Data analysis for very dynamic corpi requires...
  - High throughput updates → large batches of updates
  - Comprehensible consistency → batches applied atomically
  - Up-to-date queries → query batches before they are applied
- Pipelined DB processing self-consistent update files (SCUs)
  - Each pipeline stage produces queryable output
  - Earlier output is "fresher" but takes longer to query

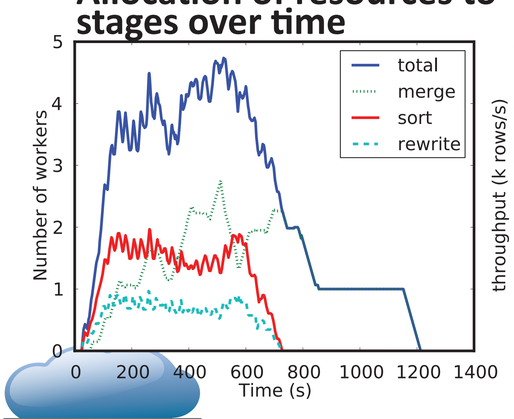




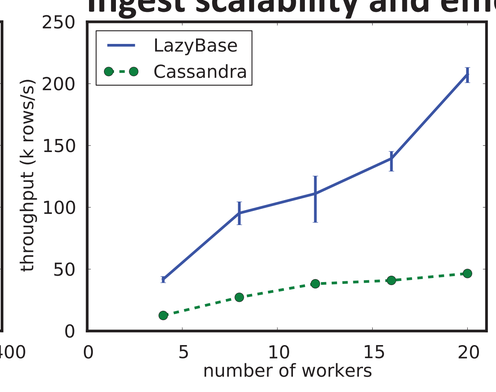
INGEST SCALING AND UTILIZATION

- Parallelizing stages solves 2 problems:
  - Need way to scale out to large clusters
  - Pipelines susceptible to bottlenecks
- Need way to assign resources to stages
  - Could model pipeline and optimize offline
    - Difficult in large heterogenous system
    - Stalled by "pig in a python" problems
  - Our approach: dynamic allocation
    - ... of stage instances and locations
    - Automatically discovers bottlenecks
    - Adjusts assignment to handle bursts

Allocation of resources to stages over time



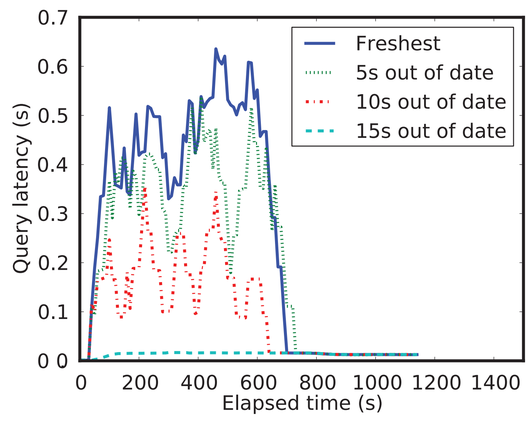
Ingest scalability and efficiency



Intel Science & Technology Center for Cloud Computing

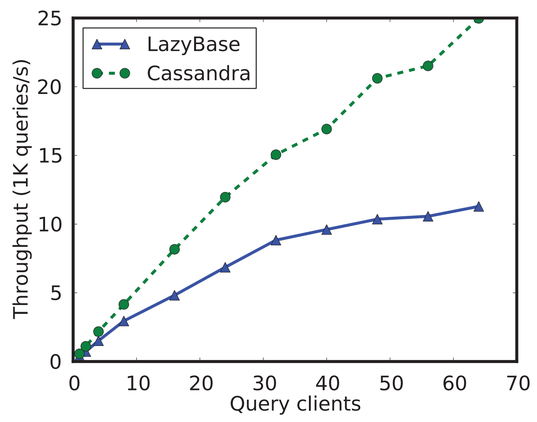
QUERY CONSISTENCY AND PERFORMANCE

- Atomic updates with consistent reads
  - Write-only transactions: most recent snapshot
  - Read-only transactions: stale consistent snapshot
- Configurable, per-query freshness
  - Expressed as seconds out-of-date
- Fresher queries examine output from earlier pipeline stages
  - Processed in parallel on all servers storing relevant data
- Data can be range partitioned for improved query parallelism



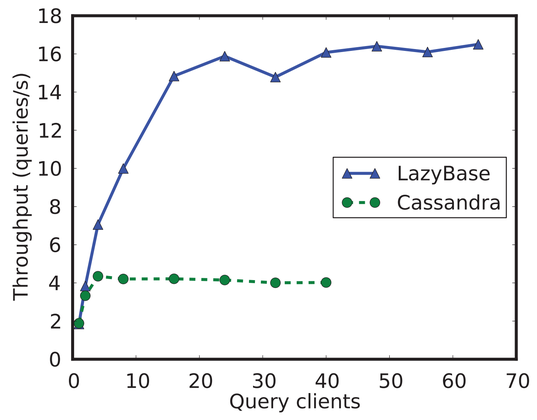
Tradeoff between freshness and query performance

- Increasing freshness also increases query latency



Point query performance

- LazyBase point query throughput limited by page decompression for



Range query performance

- LazyBase sorts data, providing dramatic improvements over Cassandra's hash distribution

FUTURE WORK

- Exploring the impact of staleness on ...
  - Consistency specification
  - Query optimization
- Matching tasks and servers intelligently
  - Get data to queryable state faster
  - Exploit data locality
  - Pipeline scheduling for improved query parallelism
- Fault tolerance
  - Intermediate data increases availability
  - Higher query cost when nodes have failed

Carnegie Mellon University · Georgia Tech · intel · PRINCETON UNIVERSITY · UC Berkeley