LazyBase: Trading Freshness for Performance in a Scalable Database
Jim Cipar, Greg Ganger, Kim Keeton (HP), Brad Morrey (HP), Craig Soules (HP), Alistair Veitch (HP)

**LazyBase**

- **Motivation:** big data analysis for very dynamic corpora
  
<table>
<thead>
<tr>
<th>Requirement</th>
<th>LazyBase design choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>High throughput ingest</td>
<td>Large batches of updates</td>
</tr>
<tr>
<td>Comprehensible consistency</td>
<td>Atomic batches applied in order</td>
</tr>
<tr>
<td>Up-to-date queries</td>
<td>Query batches before they are applied</td>
</tr>
</tbody>
</table>

- **Freshness and Performance**
  - 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  - Expressed as seconds out-of-date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

- **Query Consistency**
  - Atomic updates with consistent reads
  - Read-only transactions: stale consistent snapshot
  - LazyBase queries provide consistent but stale view of data
  - Inter-row consistency: all rows from the same snapshot
  - Monotonicity: state never goes "back in time"
  - Cassandra returns fresher, but inconsistent data

- **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
  - Freshness/performance/consistency tradeoff in other contexts
  - Geographically-distributed databases
  - Sensor networks