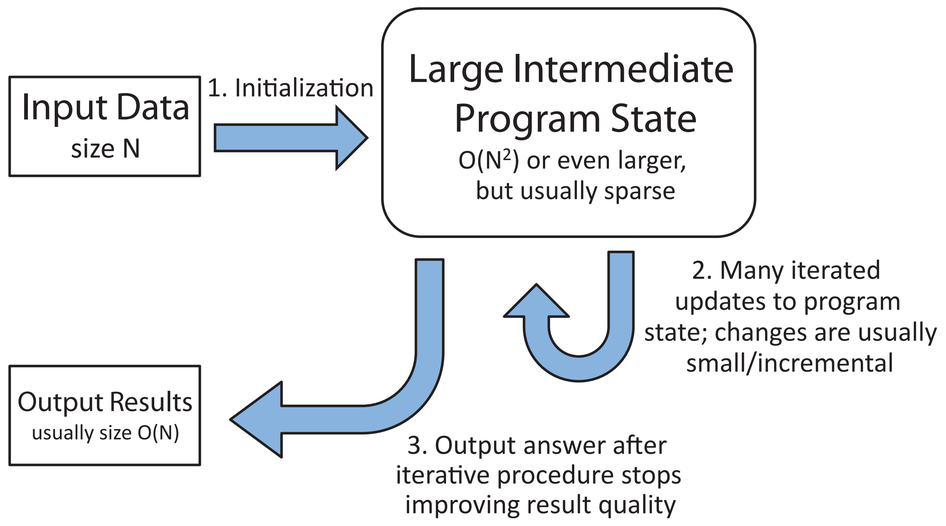
LAZYTABLES: DISTRIBUTED DATA FOR MACHINE LEARNING

Jim Cipar, Qirong Ho, Greg Ganger, Eric Xing (Carnegie Mellon University)

INTERMEDIATE ML DATA

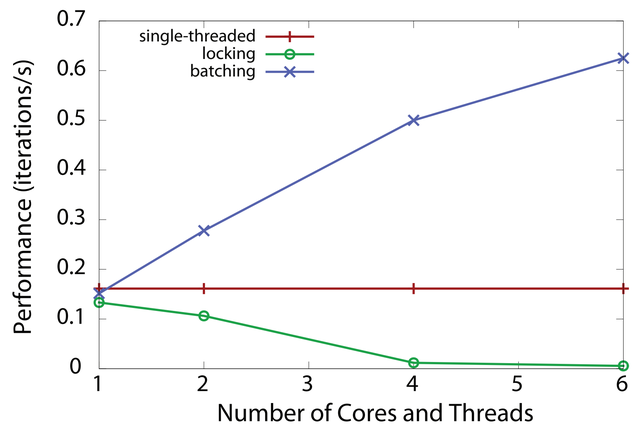
A typical ML algorithm



- Intermediate data much bigger than input and output
  - Topic modeling: 100GB input, >1TB intermediate, 10GB output
  - Usually large, sparse tables of integers
- Key problem: table performance determines efficiency
  - Hundreds of thousands of updates per second per thread
  - Too fast to lock mutex on every update
  - Self-commutative updates: increment, max, multiply...
- Key insight: algorithms tolerate staleness
  - Don't need to see other thread's updates immediately
  - But, freshness requires can grow with progress

INITIAL RESULTS

- Application tested: document classification
  - LDA algorithm implemented with Gibbs sampling
- Tested batching for 1 process on 8-core server
- "Locking" used reader-writer locks on whole table
- Batching 1024 updates in thread-local storage



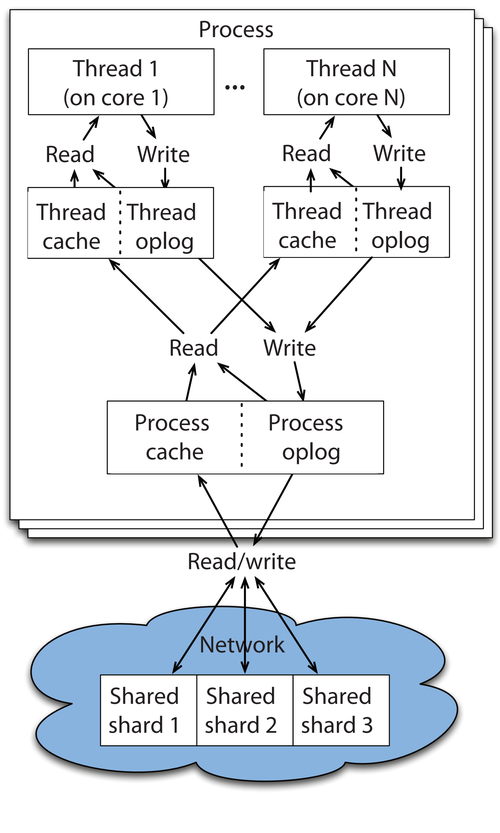
- Avoiding locks for updates is crucial to performance

CONSISTENCY

- Read-my-writes important for many algorithms
- Atomic updates to multiple rows or tables
  - Necessary for typical machine learning algorithms?
  - Can this be supported efficiently?
- When updates are sharded, enforce canonical ordering

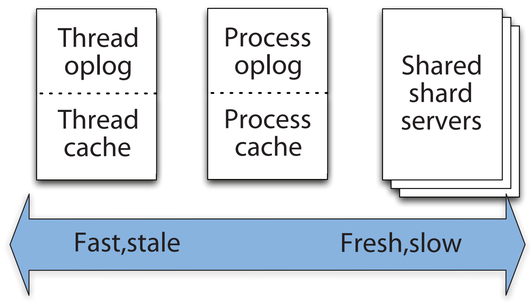
LAZYTABLES DESIGN

- Distributed table structure
- Multiple layers of caches and operation logs
  - Oplog: log of updates (e.g. "increment X by 5")
  - Thread-local, per-process, in-memory, on-disk
  - Closer layers faster but have staler data
- Write-back caching allows for update batching



SUPPORTING FRESHNESS REQUIREMENTS

- Each thread can choose cache layer for each access
- Faster caches generally hold staler data
- Oplogs hold updates from local thread or process
  - Can be ignored on read, or used for read-my-writes



- Read operations look at staleness of cache
  - If too stale, read next level cache ("freshness miss")
  - For consistency, updates flushed on freshness miss

CONTINUING EXPLORATIONS

- Testing implementation tradeoffs
  - E.g. related to consistency model
- Exploring fit to other ML algorithms
  - E.g. Image/video segmentation, community detection
- Part of larger "Big Learning" project
  - Systems support for advanced ML

Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley