Scaling Machine Learning with the Parameter Server

Mu Li, Dave Andersen, Junwoo Park, Alex Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene Shekita, Bor-Yiing Su

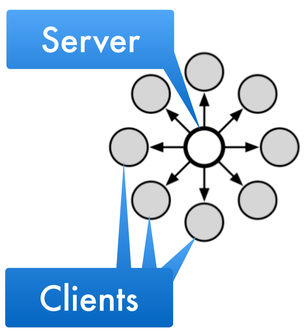
Machine Learning Problems

- Many models have O(1) blocks of O(n) terms
  (LDA, logistic regression, recommender systems)
- More features than what fits into RAM (10^{11} dimensions)
  (personalized CTR, large inventory, action space)
- Unreliable infrastructure (preemption, failure, slowdown)
- Local model typically fits into RAM
- Data needs many disks for distribution (100TB and more)
- Decouple data processing from aggregation
- Sweet spot - optimize for 80% of ML

Parameter Server

- Clients process data shards
- Clients have local view of global state and purely local state
- Parameter server has full global state
- Updates are via push / pull

Sparse Logistic Regression

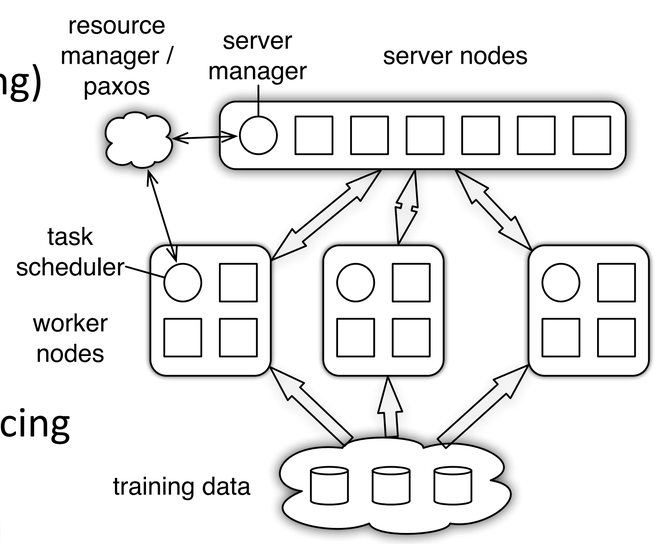


$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} \log(1 + \exp(-y_i \langle w, x_i \rangle)) + \lambda \|w\|_1$$

- **Compute gradient on (subset of data)** on each client
- **Send gradient from client to server** asynchronously
  push(key_list,value_list)
- **Proximal gradient update** on server per coordinate
- **Server returns parameters**
  pull(key_list,value_list)

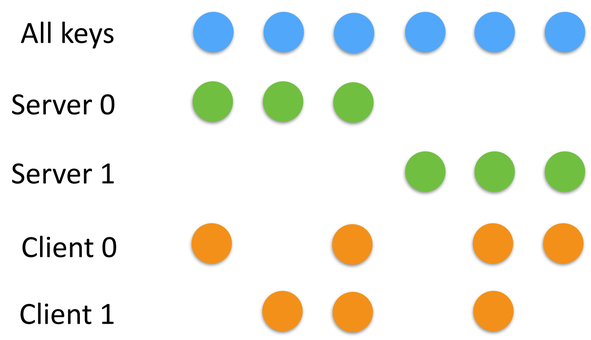
Architecture Overview

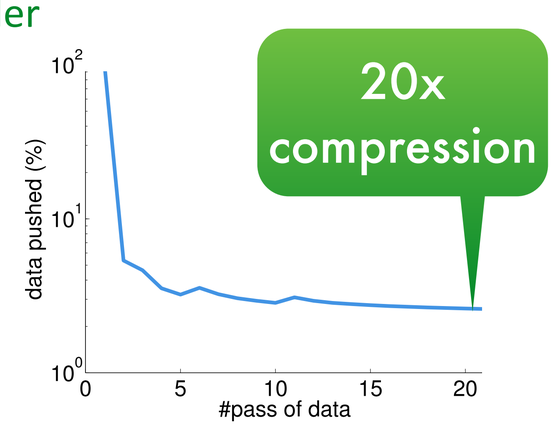
- Chord style key layout
  (keys and machines in ring)
- Replication along chain
  (a la Ouroboros)
- Recovery from failure
  by hot failover
- Multiple virtual servers
  per server for load balancing
  and efficient recovery
- Dynamic scaling for free!
- Consistency via vector clocks for ranges
- Key / value compression
- Dynamic task dependency graph and scheduler



Communication

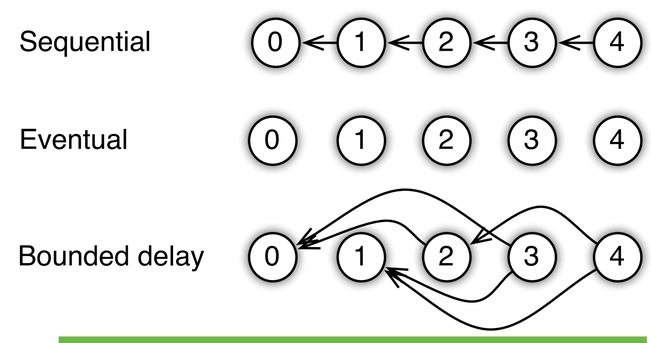
- Convergence speed depends on communication efficiency
- Sending (key,value) pairs is inefficient
  - Send only values (cache key list & checksum) instead
  - Send only (key,value) pairs that client needs
- Sending small gradients is inefficient
  - Send only sufficiently large ones instead
  - Randomize and compress accuracy of values
- Updating near-optimal values is inefficient
  - Send only large violators of KKT conditions
- Filters to allow clients / servers to self-censor
  - Avoid need for fancy scheduler





Consistency and Vector Clocks

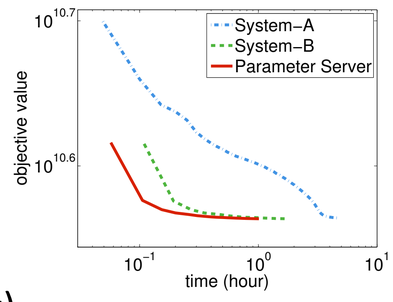
- Different consistency requirements for different models
- Use dependency graph to accomodate all of them
  (as special cases)
- Task controller sends
  subtasks to workers
- Flexible adjustment
  at runtime as needed
- Vector clocks
  - Per (key,value) pair costs
    O(clients) storage (fatal)
  - Apply for each range and
    partition only as needed



Performance

- Logistic regression (100TB CTR data)

	Method	Consistency	LOC
System-A	L-BFGS	Sequential	10,000
System-B	Block PG	Sequential	30,000
Parameter Server	Block PG	Bounded Delay KKT Filter	300



- Sketches (15 machines on 40 Gbit/s net)

Peak inserts per second	1.3 billion
Average inserts per second	1.1 billion
Peak network bandwidth per machine	4.37 GBit/s
Time to recover a failed node	0.8 second

- Topic models (4 Billion documents, 60k cores, 1M tokens)

Intel Science & Technology Center for Cloud Computing

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