More Effective Distributed ML with a stale synchronous parallel parameter server


http://www.istc-cc.cmu.edu/
Machine Learning is Ubiquitous

Learning of graphical models

- Nonparametric Bayesian inference
- Nonstationary time series analysis
- Multi-task Learning & structured I/O models
- Theory of nonparametric high-dimensional inference

Applications:

1. Social networks and social media
2. Computational biology, and genomics
3. Web-scale text mining and NLP
4. Anomaly detection and video surveillance
5. Computational finance
6. Web-scale image mining

ACGTTTTACTGTACAATT
ACGTTTTACTGTACAATT

ACGTTTTACTGTACAATT
ACGTTTTACTGTACAATT
Challenge #1 - Massive Data Scale

Facebook 2012

The latest on everybody's favorite social network

1 of every 5 of all page views
845 million monthly active users
100 billion connections

~1B nodes, do not fitting into the main memory of a single machine, a familiar problem!

Source: Internet Systems Consortium (www.isc.org)
\[ \arg\max_{\beta} \mathcal{L}(\{x_i, y_i\}; \beta) + \Omega(\beta) \]

\( > 10^{11} \) parameters, do not fitting into the main memory of a single machine either!
A Thin Waist?

Machine Learning Families
- Graphical Models
- Nonparametric Bayesian Models
- Regularized Bayesian Methods
- Large-Margin
- Sparse Structured I/O Regression
- Sparse Coding
- Spectral/Matrix Methods
- Others

Algorithmic Building Blocks

Hardware and infrastructure
- Network switches
- Infiniband
- Network attached storage
- Flash storage
- Server machines
- Desktops/Laptops
- GPUs
- NUMA machines
- Cloud compute (e.g. Amazon EC2)
- Virtual Machines

© Eric Xing @ CMU, 2013
Toward A General-Purpose Framework for Big Learning

**Theory:** Degree of parallelism, convergence analysis, sub-sample complexity ...

**Representation:** Compact and informative features

**Model:** Generic building blocks: loss functions, structures, constraints, priors ...

**Algorithm:** Parallelizable and stochastic MCMC, VI, Opt, Spectrum ...

**System:** Distributed architecture: DFS, parameter server, task scheduler ...

**Hardware:** GPU, flash storage, cloud ...

© Eric Xing @ CMU, 2013
An Example Task: Topical Inference

- Say, we want to have a mapping ..., so that

  - Compare similarity
  - Classify contents
  - Cluster/group/categorize docs
  - Distill semantics and perspectives
  - ...

![Diagram with three topics: Politics, Economics, Sports]
Markov Chain Monte Carlo:
Randomly sample each variable in sequence

Stochastic Variational Inference:
Gradient ascent on randomly-chosen variables
The Need for Distributed Computation

- Ex: Topic Modeling MCMC samplers process 100K-1M words per second for 100 topics, on an 8-core machine
  - If one document is 1000 words, that’s 100-1000 docs per second

- What if we want 1B docs and 10K topics?

- Memory:
  - 1B * 10K = 10 trillion parameters = 40TB of RAM

- Computation:
  - 1B docs, 10K topics -> 100M seconds on one machine -> 1000+ days!

- Need many machines for memory, computational requirements!
Distributed Sampling Cycle

Sample \( \Omega_t \)
Requires a reduction step
Parameter Servers for ML

- Provide simple table-based API for quick porting of single-machine parallel ML programs
  - `read_row(table,row,s)`
    - Retrieve table-row with staleness s
  - `inc(table,row,el,val)`
    - Increment table(row,el) by val
  - `clock()`
    - Inform all servers that current/thread processor has completed one clock

- ML programmers simply replace array/matrix data structures with parameter server calls

- Question: what synchronizing scheme to use?
Synchronization Models for PS

- Bulk Synchronous Parallel execution
  - No update errors, but synchronization barriers introduce stragglers and waste computational time

- Asynchronous execution
  - No stragglers, but update errors can end up unbounded in distributed settings (e.g. one machine is systematically slower than the rest)
• ML programs are **iterative-convergent**
  - Repeatedly execute update equations to minimize a loss function
  - Examples:
    - Variational inference for topic models
    - Stochastic gradient descent for matrix factorization
    - MCMC to find posterior distribution modes
    - Block coordinate descent for Lasso regression

• **Iterative-convergent algorithms are empirically and theoretically resilient to errors in updates**
  - Errors will decrease update effectiveness, but will not forfeit convergence and correctness provided they are limited
Stale Synchronous Parallelism (SSP)

Allow threads to run at their own pace, without synchronization
Ensure fastest/slowest threads do not grow more than $S$ iterations apart
Serve data from thread-local and process-local caches rather than over the network
• **Bulk Synchronous Parallel execution**
  ▫ No update errors, but synchronization barriers introduce stragglers and waste computational time

• **Asynchronous execution**
  ▫ No stragglers, but update errors can end up unbounded in distributed settings (e.g. one machine is systematically slower than the rest)

• **Stale Synchronous Parallel (SSP)**
  ▫ Use bounded staleness to strictly limit maximum error, while reducing synchronization costs
  ▫ Aims for a sweet spot between BSP and Async
Cache hierarchy for staleness

- **LazyTables Parameter Server**
  - Stale reads served by local thread/process caches on the client machine
  - Only read from server if local caches are too stale
  - Writes are immediately committed after each clock()
    - Okay since ML programs perform far more reads than writes
Enjoys async speed, but BSP guarantee

**LDA on NYtimes Dataset**
LDA 32 machines (256 cores), 10% docs per iter

SSP is both fast and has convergence guarantees
BSP has convergence guarantees but is slow
Full Asynchronous is fast but has weak convergence guarantees
LDA on NYtimes dataset
(staleness = 10, 1k docs per core per iteration)

Doubling # machines almost halves time taken to converge

SSP computational model enables linear scaling with # machines (up to at least 32)
Network bottlenecks in ML Mitigated

**Time Breakdown: Compute vs Network**
LDA 32 machines (256 cores), 10% data per iter

Network communication is a large bottleneck with 10s of machines
**SSP balances network and compute time**
Quality vs Quantity tradeoff

**Quantity: iterations versus time**  
LDA 32 machines, 10% data

**Quality: objective versus iterations**  
LDA 32 machines, 10% data

Progress per time is (iters/sec) * (progress/iter)
High staleness yields more iters/secs, but lowers progress/iter

Find the sweet spot staleness >0 that yields maximum progress per time
Effective across different ML Programs

Matrix Factorization on Netflix

Objective function versus time
MF 32 machines (256 threads)

Lasso on synthetic data

Objective function versus time
Lasso 16 machines (128 threads)
Why SSP converges despite error

When a thread reads a parameter, the number of “missing updates” is bounded (compared to sequential execution)
Hence numeric error in parameter is also bounded
**Why SSP converges despite error**

**Theorem 1 (SGD under SSP):** Suppose we want to find the minimizer $x^*$ of a convex function $f(x) = \frac{1}{T} \sum_{t=1}^{T} f_t(x)$, via gradient descent on one component $\nabla f_t$ at a time. We assume the components $f_t$ are also convex. Let $u_t := -\eta_t \nabla f_t(\hat{x}_t)$, where $\eta_t = \frac{\sigma}{\sqrt{t}}$ with $\sigma = \frac{F}{L\sqrt{2(s+1)P}}$ for certain constants $F, L$. Then, under suitable conditions ($f_t$ are $L$-Lipschitz and the distance between two points $D(x||x') \leq F^2$),

$$R[X] := \left[ \frac{1}{T} \sum_{t=1}^{T} f_t(\hat{x}_t) \right] - f(x^*) \leq 4FL \sqrt{\frac{2(s+1)P}{T}}$$

**SSP converges for Stochastic Gradient Descent**, and does so at a rate no slower than $O(T^{-0.5})$, where $T$ is the number of iterations. This rate is only an upper bound - depending on how the parameter server is implemented, there is room to make the actual convergence rate approach the optimal rate for a single core.
• **What is an ML program:** An objective or **loss function** that measures solution quality
  
  - e.g. least-squared loss in regression
  - e.g. negative log-likelihood in probabilistic models
  - e.g. risk in adversarial or bandit problems

• **Key Signatures of an ML program**
  
  ▫ **ML algorithms are iterative-convergent**
    • Iterative-convergent algorithms are **resilient to errors in updates**

  ▫ **ML programs are blocky**
    • Opportunities for efficient **parallelization under bounded error**

• **A general Big ML Framework shall leverage these properties**

© Eric Xing @ CMU, 2013
The Big-ML Framework We Envision

Algorithmic Building Blocks
- Distributed MC
- Graph Propagation
- Convex Optimization
- Spectral Algorithms
- Stochastic Inference

Machine Learning Families
- Graphical Models
- Nonparametric Bayesian Models
- Regularized Bayesian Methods
- Large-Margin
- Sparse Structured I/O Regression
- Sparse Coding
- Spectral Matrix Methods
- Others

System Building Blocks
- Big Model System
  - Dynamic Scheduling
  - Adaptive Load-Balancing
  - Client Autonomicity
- Programming Interface
  - For ML practitioners
  - For ML scientists
  - APIs for Power users
- Big Data System
  - Data Partitioning
  - Parameter Server
  - Thread-Level Caching
  - Multi-instance tenancy

Hardware and infrastructure
- Network switches
- Network attached storage
- Server machines
- GPUs
- Desktops/Laptops
- NUMA machines
- Cloud compute (e.g. Amazon EC2)
- Virtual Machines

“Thin Waist”
Thank You!