A Systems Approach to Scalable Bayesian Inference

Joseph E. Gonzalez
Postdoc, UC Berkeley AMPLab
Co-founder, GraphLab Inc.
jegonzal@eecs.berkeley.edu
Data Velocity is an opportunity for Bayesian Nonparametrics.

How do we scale Bayesian inference?
Opposing Forces

Accuracy

Ability to estimate the posterior distribution

Serial Inference

Scalability

Ability to effectively use parallel resources

Parameter Server

Coordination Free Samplers
Serial Inference

Data

Model
State

Serial Inference
Coordination Free Parallel Inference
Coordination Free Parallel Inference

Keep Calm and Carry On.
Parameter Servers
System for Coordination Free Inference


A. Smola and S. Narayananmurthy. An architecture for parallel topic models. VLDB’10


Ho et al. “More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server.” NIPS’13
Hierarchical Clustering

Global Variables

\[ \mu_j \]

\[ j \in \{1, \ldots, K\} \]

Local Variables

\[ x_i \]

\[ i \in \{1, \ldots, N\} \]
**Example: Topic Modeling with LDA**

- **Word Dist. by Topic**
  - \( \beta_t \)
  - \( t \in \{1, \ldots, T\} \)
  - Maintained by the Parameter Server

- **Local Variables Documents**
  - \( x_i \)
  - \( z_i \)
  - \( \theta_d \)
  - Tokens
  - \( i \in \{1, \ldots, \text{Len}(d)\} \)
  - \( d \in \{1, \ldots, D\} \)
  - Maintained by the Workers Nodes

- **Maintained by the**
  - **Parameter Server**
  - **Workers Nodes**
Ex: Collapsed Gibbs Sampler for LDA

Partitioning the model and data

Parameter Server

$W_{1:10K}$

Parameter Server

$W_{10k:20K}$

Parameter Server

$W_{20k:30K}$
Ex: Collapsed Gibbs Sampler for LDA

Partitioning the model and data

Parameter Server

Parameter Server

Parameter Server

Parameter Cache

Parameter Cache

Parameter Cache

Parameter Cache
Ex: Collapsed Gibbs Sampler for LDA

Inconsistent model replicas

Parameter Server

\(W_{1:10K}\)

\(W_{10k:20K}\)

\(W_{20k:30K}\)

Parameter Cache

Car
Dog
Bat
Pig
Cat
Gas
VW
Zoo
Car
Rim
$$
bmw
Mac
iOS
iPod
Parallel Gibbs Sampling

Incorrect Posterior

dependent variables cannot in general be sampled simultaneously.
Issues with Nonparametrics

Difficult to introduce new clusters asynchronously:

Leads to too many clusters!
Opposing Forces

Accuracy

Serial Inference

Scalability

Parameter Server

Asynchronous Samplers
Opposing Forces

Accuracy

Scalability

Accuracy

Scalability
Opposing Forces

Accuracy

Serial Inference

Scalability

Parameter Server

Asynchronous Samplers
Opposing Forces

Accuracy

Scalability

Serial Inference

Concurrency Control

Parameter Server

Asynchronous Samplers
Concurrency Control

Coordination Free (Parameter Server):

Provably fast and correct under key assumptions.

Concurrency Control:

Provably correct and fast under key assumptions.
Opposing Forces

Concurrency Control

Serial Inference

Mutual Exclusion

Optimistic Concurrency Control

Safe

Unsafe

Parameter Server

Asynchronous Samplers

Accuracy

Scalability
Mutual Exclusion

Conditional Independence


Exploit the Markov random field for

*Parallel Gibbs Sampling*

Graph Coloring

R/W Lock Mutual Exclusion
Mutual Exclusion through Scheduling
Chromatic Gibbs Sampler

Compute a k-coloring of the graphical model

Sample all variables with same color in parallel

Serial Equivalence:
Theorem: *Chromatic Sampler*

**Ergodic:** converges to the correct distribution
- Based on graph coloring of the Markov Random Field

**Quantifiable** acceleration in mixing

\[ \text{Time to update all variables once} \approx \frac{n}{p} + k \]

- \# Variables
- \# Colors
- \# Processors
Introducing locking (scheduling) protocols to identify potential conflicts.
Mutual Exclusion Through Locking

Enforce serialization of computation that could conflict.
Markov Blanket Locks

Read/Write Locks:
Markov Blanket Locks

Eliminate fixed schedule and global coordination

Supports more advanced block sampling

Expected Parallelism:

\[ E(\text{#active processors}) \geq 1 + (p - 1) \left( 1 - (p - 1) \left( \frac{d + 1}{n} \right) \right) \]

# Processors \quad # Variables

Max Degree
A System for Mutual Exclusion on Markov Random Fields

GraphLab/PowerGraph [UAI’10, OSDI’12]:

• Chromatic Sampling
• Markov Blanket Locks + Block Sampling
Limitation
Densely Connected MRF

V-Structures: observations couple many variables

Collapsed models: clique-like MRFs

Mutual exclusion *pessimistically* serializes computation that *could* interfere.

Can we be *optimistic* and only serialize computation that *does* interfere?
Opposing Forces

- Serial Inference
- Mutual Exclusion
- Optimistic Concurrency Control

Accuracy

Scalability

Parameter Server

Unsafe
Safe
Optimistic Concurrency Control

assume the best and correct


Xinghao Pan
Tamara Broderick
Stefanie Jegelka
Michael Jordan
Optimistic Concurrency Control

Classic idea from Database Systems:

ACM Transactions on Database Systems. 1981

Assume most operations won’t conflict:

- Execute operations without blocking
  
  *Frequent case is fast*

- Identify and resolve conflicts *after* they occur
  
  *Infrequent case with potentially costly resolution*
Allow computation to proceed without blocking.

*ACM Transactions on Database Systems* 1981
Optimistic Concurrency Control

Validate potential conflicts.

Optimistic Concurrency Control

Validate potential conflicts.

Kung & Robinson. On optimistic methods for concurrency control.
ACM Transactions on Database Systems 1981
Optimistic Concurrency Control

Model State

Data

Processor 1

Processor 2

Take a compensating action.

Optimistic Concurrency Control

Model State

Data

Validating potential conflicts.

Optimistic Concurrency Control

Take a compensating action.

ACM Transactions on Database Systems 1981
Optimistic Concurrency Control

Requirements:
- **Fast** → Validation: *Identify Errors*
- **Infrequent** → Resolution: *Correct Errors*
- **Non-Blocking Computation** → **Concurrency**

Model State

Rollback and Redo

Data
Optimistic Concurrency Control for Bayesian Inference

Non-parametric Models [Pan et al., NIPS’13]:

- OCC DP-Means: Dirichlet Process Clustering
- OCC BP-Means: Beta Process Feature Learning

Conditional Sampling: (In Progress)

- Collapsed Gibbs LDA
- Retrospective Sampling for HDP
DP-Means Algorithm

[Kulis and Jordan, ICML’12]

Start with DP Gaussian mixture model:

\[ G \sim \text{DP} \left( \alpha, G_0 = \mathcal{N}(0, \rho I) \right) \]

\[ \phi_i \sim G \]

\[ x_i \sim \mathcal{N}(\phi_i, \sigma I) \]

small variance limit \( \sigma \rightarrow 0 \)
DP-Means Algorithm

[Kulis and Jordan, ICML’12]

Start with DP Gaussian mixture model:

\[ G \sim \text{DP} \left( \alpha, G_0 = \mathcal{N}(0, \rho I) \right) \]
\[ \phi_i \sim G \]
\[ x_i \sim \mathcal{N}(\phi_i, \sigma I) \]

small variance limit \( \sigma \rightarrow 0 \) redefine \( \alpha \): 

\[ \alpha(\sigma) = \left( 1 + \frac{\rho}{\sigma} \right)^{d/2} \exp \left( -\frac{\lambda}{2\sigma} \right) \]

Decreases Rapidly
DP-Means Algorithm

[Kulis and Jordan, ICML’12]

Corresponding Gibbs sampler conditionals:

\[ P(\text{join } c) \propto n_{-i,c} \exp \left( -\frac{1}{2\sigma} \left[ \lambda \left( \frac{\sigma}{\rho + \sigma} \|x_i\|^2 \right) \right] \right) \]

\[ P(\text{new}) \propto \exp \left( -\frac{1}{2\sigma} \left[ \lambda \left( \frac{\sigma}{\rho + \sigma} \|x_i\|^2 \right) \right] \right) \]

Taking the small variance limit  \( \sigma \to 0 \)
DP-Means Algorithm

[Kulis and Jordan, ICML’12]

Gibbs updates become deterministic:

for \( i \in \{1, \ldots, n\} \) do

\[ c_{i}^{\text{min}} = \arg \min_{c} \| x_{i} - \mu_{c} \| \]

\[ \begin{align*}
\text{if} \quad & \| x_{i} - \mu_{c_{i}^{\text{min}}} \| < \lambda \\
\text{else} & \quad \text{create new cluster at} \quad \mu_{k+1} = x_{i}
\end{align*} \]

Taking the small variance limit \( \sigma \to 0 \)
Gibbs updates become deterministic:

\[
\text{for } i \in \{1, \ldots, n\} \text{ do } \\
\quad c_i^{\text{min}} = \arg \min_c \|x_i - \mu_c\| \\
\quad \text{if } \|x_i - \mu_{c_i^{\text{min}}}\| < \lambda \text{ then } \text{join } c_i^{\text{min}} \\
\quad \text{else } \text{create new cluster at } \mu_{k+1} = x_i \\
\text{for } c \text{ in clusters do } \mu_c \leftarrow \frac{1}{n_c} \sum_{x \in c} x
\]
DP-Means Algorithm

[Kulis and Jordan, ICML’12]

Computing cluster membership
DP-Means Algorithm

[Kulis and Jordan, ICML’12]

Updating cluster centers:
DP-Means Parallel Execution

Computing cluster membership in parallel:

Cannot introduce overlapping clusters in parallel
Optimistic Concurrency Control for Parallel DP-Means

Optimistic Assumption
No new cluster created nearby

Validation
Verify that new clusters don’t overlap

Resolution
Assign new cluster center to existing cluster
OCC DP-means

**Theorem:** OCC DP-means is serializable and therefore preserves theoretical properties of DP-means.

**Theorem:** Assuming well spaced clusters the expected overhead of OCC DP-means does not depend on data size.
Empirical Validation Failure Rate

![Graph showing the relationship between dataset size, separable clusters, and points failing validation across different numbers of processors. The graph illustrates how the failure rate changes with respect to dataset size, with distinct lines for each processor count, ranging from 2 to 32 processors. The x-axis represents the dataset size, while the y-axis shows the points failing validation. The graph also highlights the impact of OCC overhead on parallelism.](image-url)
Empirical Validation Failure Rate

Overlapping Clusters

Points Failing Validation vs. Dataset Size

- 2 Processors
- 4 Processors
- 8 Processors
- 16 Processors
- 32 Processors

OCC Overhead vs. Parallelism

- \( P_b = 16 \)
- \( P_b = 32 \)
- \( P_b = 64 \)
- \( P_b = 128 \)
- \( P_b = 256 \)
Distributed Evaluation Amazon EC2

~140 million data points; 1, 2, 4, 8 machines

Runtime in Second
Per Complete Pass over Data

Number of Machines

OCC DP-means Runtime
Projected Linear Scaling

2x #machines
≈ ½x runtime
Optimistic Future for Optimistic Concurrency Control

Optimistic Concurrency Control

Accuracy

Serial Inference

Scalability

Parameter Server

Unsafe

Safe
Thank You

Questions?

Joseph E. Gonzalez

ejgonzal@eecs.berkeley.edu
OCC Collapsed Gibbs for LDA

Maintain epsilon intervals on the conditional:

System ensures conditionals are $\epsilon$-accurate

Validate: Accept draws that land outside interval

Resolution: Serially resample rejected tokens
OCC Collapsed Gibbs for LDA

**Ratio of runtime of OCC LDA to Hogwild LDA**

**Speed up of multicore LDA**
Optimism for Optimistic Concurrency Control

Validation

Frequent
Fast + Easy
Coordination Free

Rare
Slow + Complex
Resolution

Enable decades of work in serial Bayesian inference to be extended to the parallel setting.