Scalable Inference In Latent Variable Models

Amr Ahmed , Mohamed Aly, Joseph Gonzalez, Shravan Narayanamurthy, Alexander Smola

Yahoo! Research & CMU

Motivations: Data is everywhere

with labels

- Ads
- Click Feedback
- Emails
- Tags
- Editorial data is very expensive! Do not use!

Unlabeled

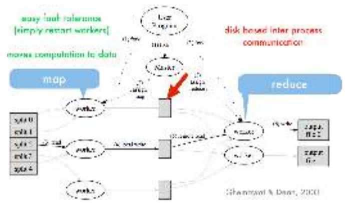
- Graphs
- Document collections
- Email/IM/Discussions
- Query stream

•Essentially infinite amount of data
•Labeling is prohibitively
•Even for *supervised problems unlabeled data abounds.*
•*User-understandable structure for representation purposes*
•*Solutions are often customized to problem*

Challenges

•Millions to billions of instances
•Rich structure of data (ontology, categories, tags)
•Model description typically larger than memory of a workstation
•Usually clustering or topic models do not solve the problem
•Temporal structure of data
•Side information for variables
•10k-100k clusters for hierarchical model
•1M-100M words
•Communication is an issue for large state space

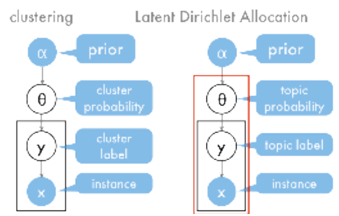
Map-Reduce is not the solution



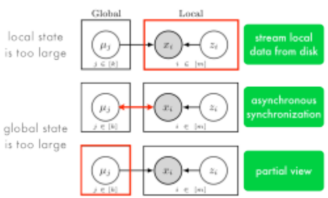
•Good if only a small number of MapReduce iterations needed
•Need to request machines at each iteration (time consuming)
•State lost in between maps
•Communication only via file I/O

Not a good fit for many latent variable models which are iterative in nature and relies on a shared state

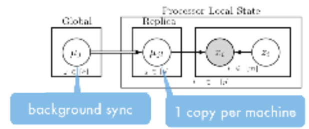
Examples:



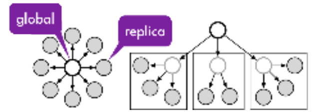
Three Basic inference problems:

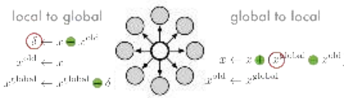


Global State Synchronization



- No locks between machines to access z
- Synchronization mechanism for global μ needed
- In LDA this is the local copy of the (topic,word) counts

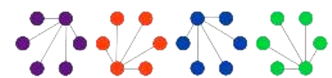




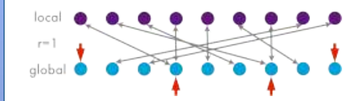
•Start with common state
•Child stores old and new state
•Parent keeps global state
•Transmit differences asynchronously
  - Inverse element for difference
  - Abelian group for commutativity
  (sum, log-sum, cyclic group, exponential families)

Key distribution and Fault Tolerance

$$m(x) = \operatorname*{argmin}_{m \in M} h(x, m)$$



•Dedicated server for variables
•Select server via consistent hashing
•Storage is O(1/k) per machine
• Communication is O(1) per machine
•Fast snapshots O(1/k) per machine
  (stop sync and dump state per vertex)



•Schedule message pairs
•Communicate with r random machines simultaneously
•Use Luby-Rackoff PRPG for load balancing
•Efficiency guarantee:

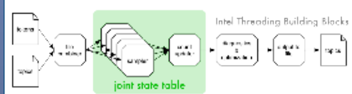
$$1 - e^{-r}\sum_{i=0}^{r}\left[1-\frac{i}{r}\right]\frac{r^i}{i!} \le \text{Eff} \le 1 - e^{-r}$$

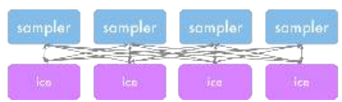
4 simultaneous connections are sufficient

Architecture: LDA

- For 1000 iterations do (independently per computer)
  - For each thread/core do
    - For each document do
      - For each word in the document do
        - Resample topic for the word
        - Update local (document, topic) table
        - Generate computer local (word, topic) message
    - In parallel update local (word, topic) table
  - In parallel update global (word, topic) table

concurrent cpu hdd net | minimal view | continuous sync | barrier free

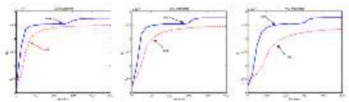




• Distributed (key,value) storage via ICE
•Background asynchronous synchronization
  - single word at a time to avoid deadlocks
  - no need to have joint dictionary
  - uses disk, network, cpu simultaneously

Results

•8 Million documents, 1000 topics, {100,200,400} machines, LDA
• Red (symmetric latency bound message passing)
• Blue (asynchronous bandwidth bound message passing & message scheduling)
•10x faster synchronization time and 10x faster snapshots
•Scheduling improves 10% already on 150 machines



Applications: Temporal Models