GraphLab: A New Framework for Parallel Machine Learning

Yucheng Low, Aapo Kyrola, Carlos Guestrin,
Joseph Gonzalez, Danny Bickson, Joe Hellerstein

Presented by Guozhang Wang

DB Lunch, Nov.8, 2010

Overview

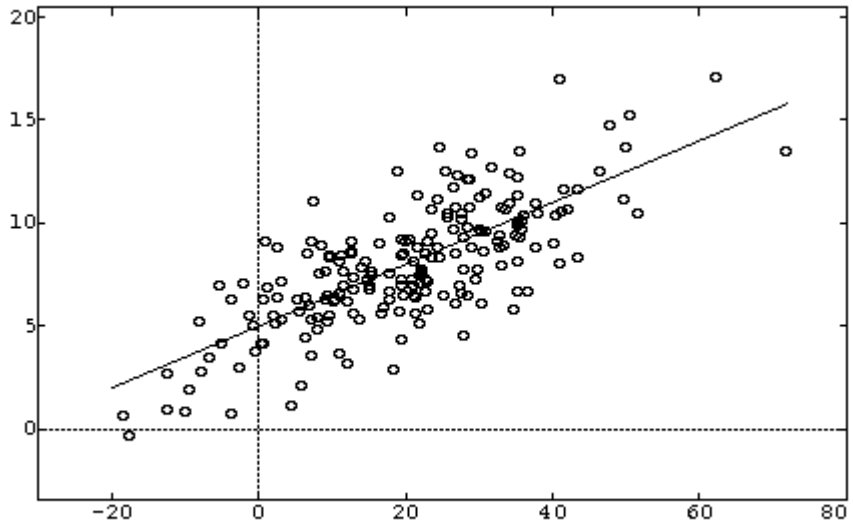
- Programming ML Algorithms in Parallel
  - Common Parallelism and MapReduce
  - Global Synchronization Barriers
- GraphLab
  - Data Dependency as a Graph
  - Synchronization as Fold/Reduce
- Implementation and Experiments
- From Multicore to Distributed Environment

Parallel Processing for ML

- Parallel ML is a Necessity
  - 13 Million Wikipedia Pages
  - 3.6 Billion photos on Flickr
  - etc
- Parallel ML is Hard to Program
  - Concurrency v.s. Deadlock
  - Load Balancing
  - Debug
  - etc

MapReduce is the Solution?

- High-level abstraction: Statistical Query Model [Chu et al, 2006]

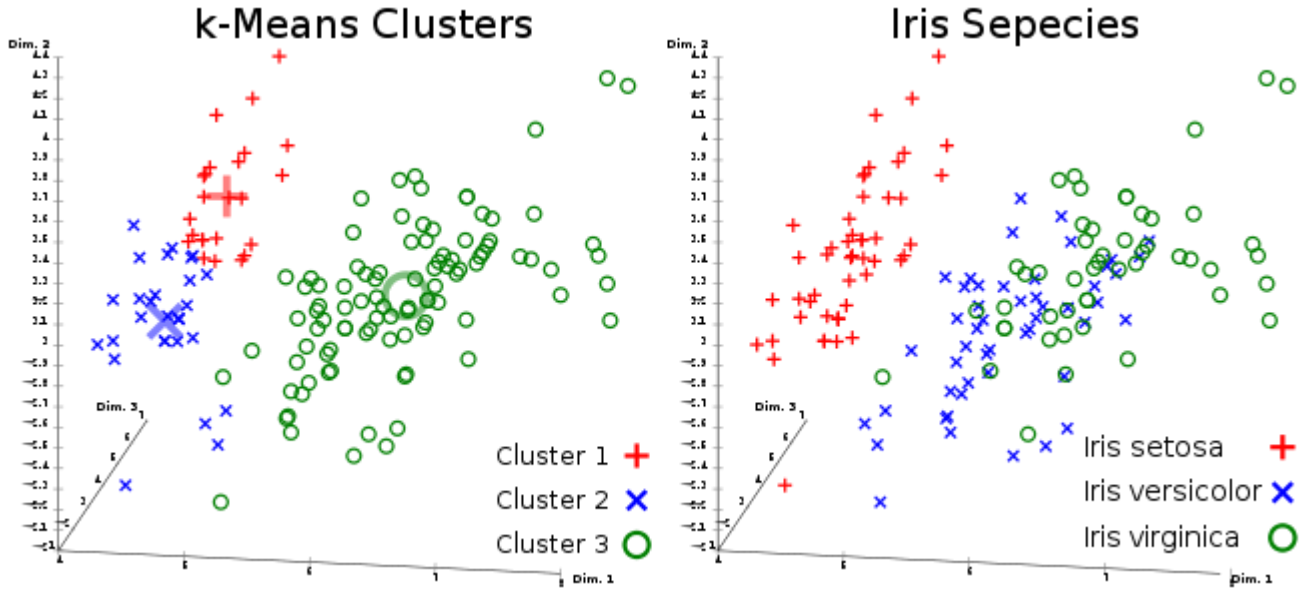


Weighted Linear Regression: only sufficient statistics

$$\theta = A^{-1}b,\ A = \Sigma w_i(x_i x_i^T),\ b = \Sigma w_i(x_i y_i)$$

MapReduce is the Solution?

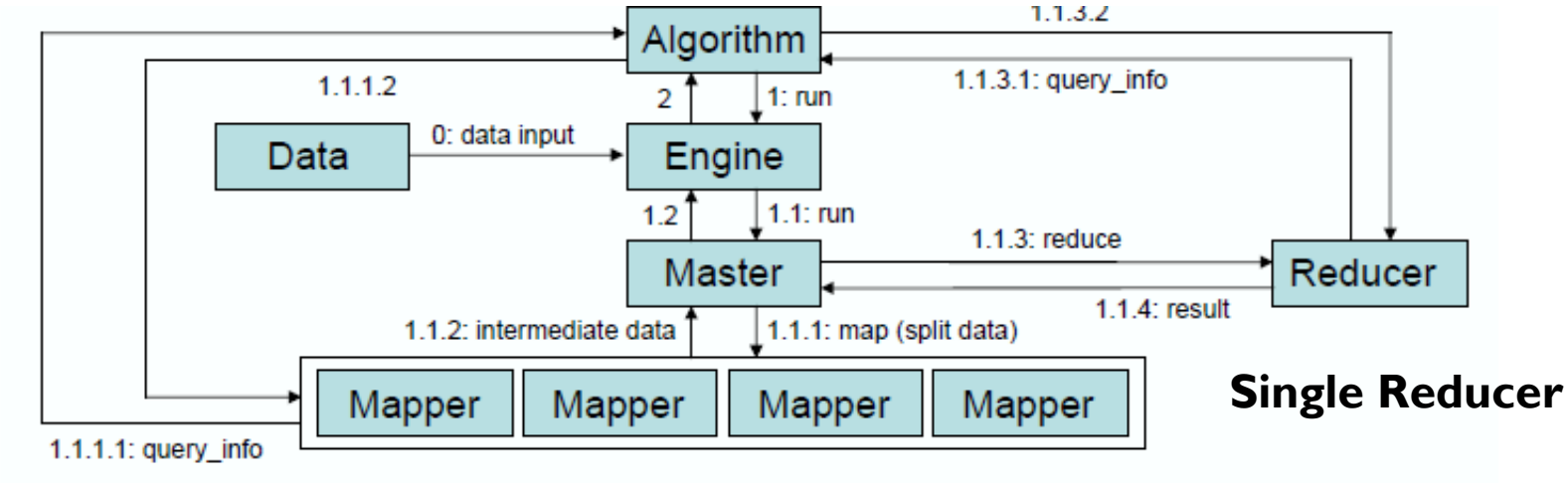
- High-level abstraction: Statistical Query Model [Chu et al, 2006]



Embarrassingly Parallel independent computation

No Communication needed

ML in MapReduce



- Iterative MapReduce needs global synchronization at the single reducer
  - K-means
  - EM for graphical models
  - gradient descent algorithms, etc

Not always Embarrassingly Parallel

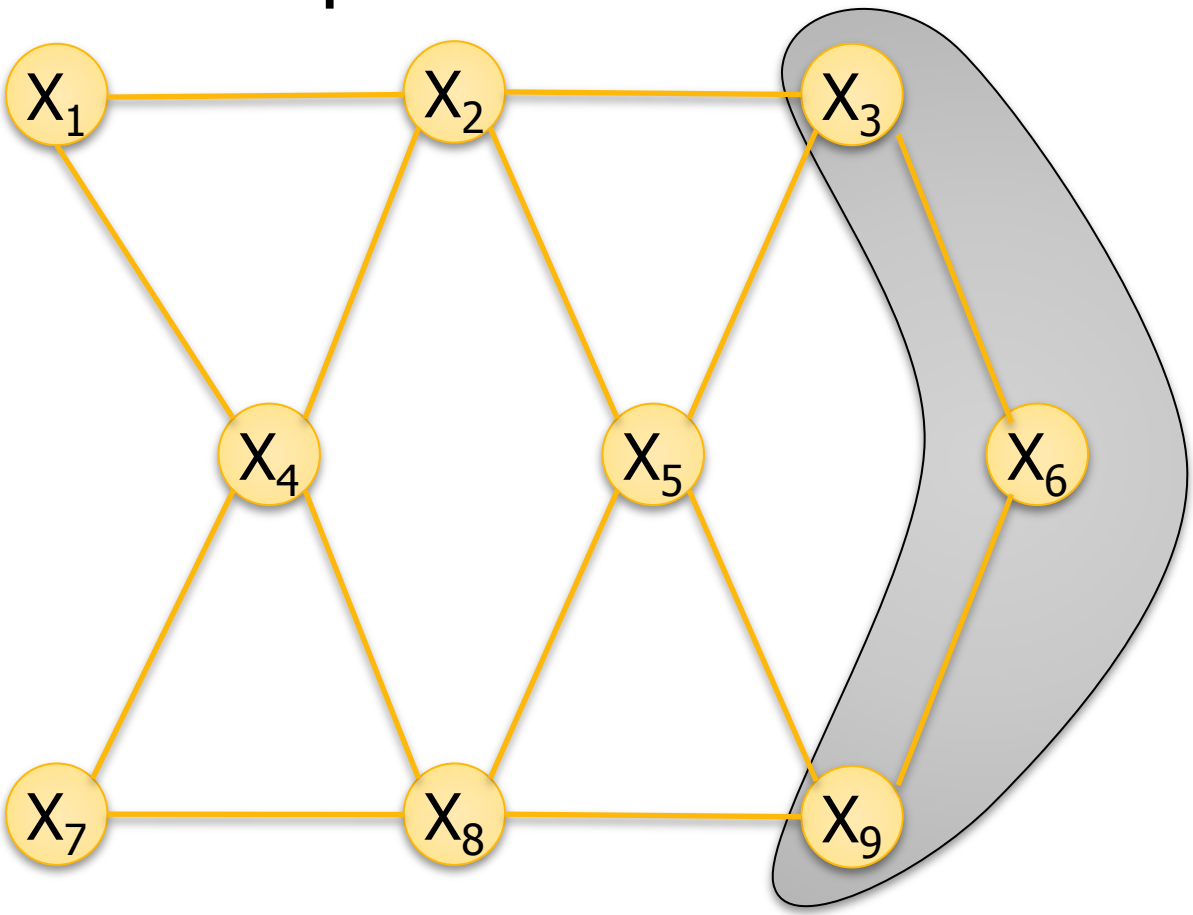
- Data Dependency: not MapReducable
  - Gibbs Sampling
  - Belief Propagation
  - SVM
  - etc
- Capture Dependency as a Graph!

Overview

- Programming ML Algorithms in Parallel
  - Common Parallelism and MapReduce
  - Global Synchronization Barriers
- GraphLab
  - Data Dependency as a Graph
  - Synchronization as Fold/Reduce
- Implementation and Experiments
- From Multicore to Distributed Environment

Key Idea of GraphLab

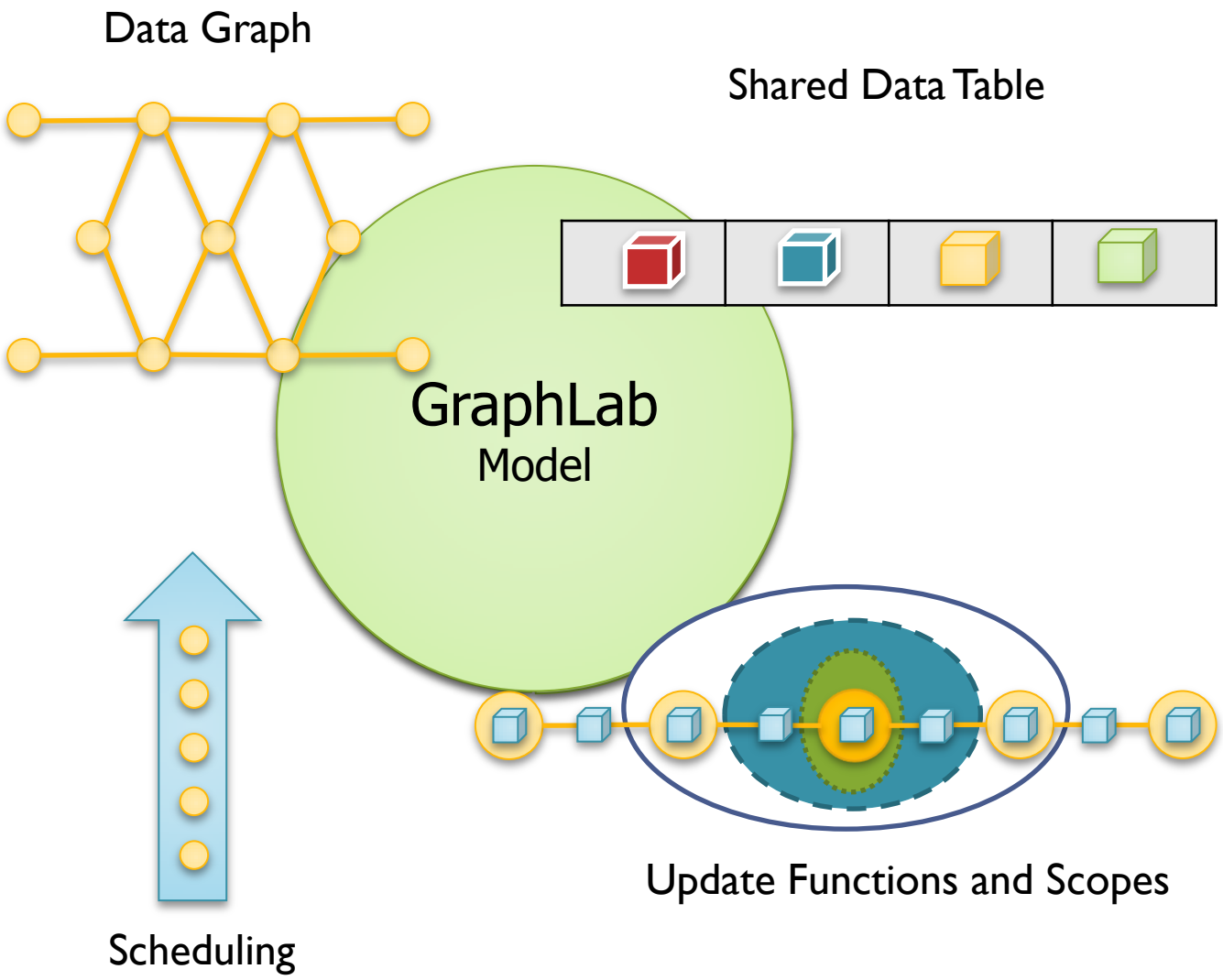
- Sparse Data Dependencies
- Local Computations



GraphLab for ML

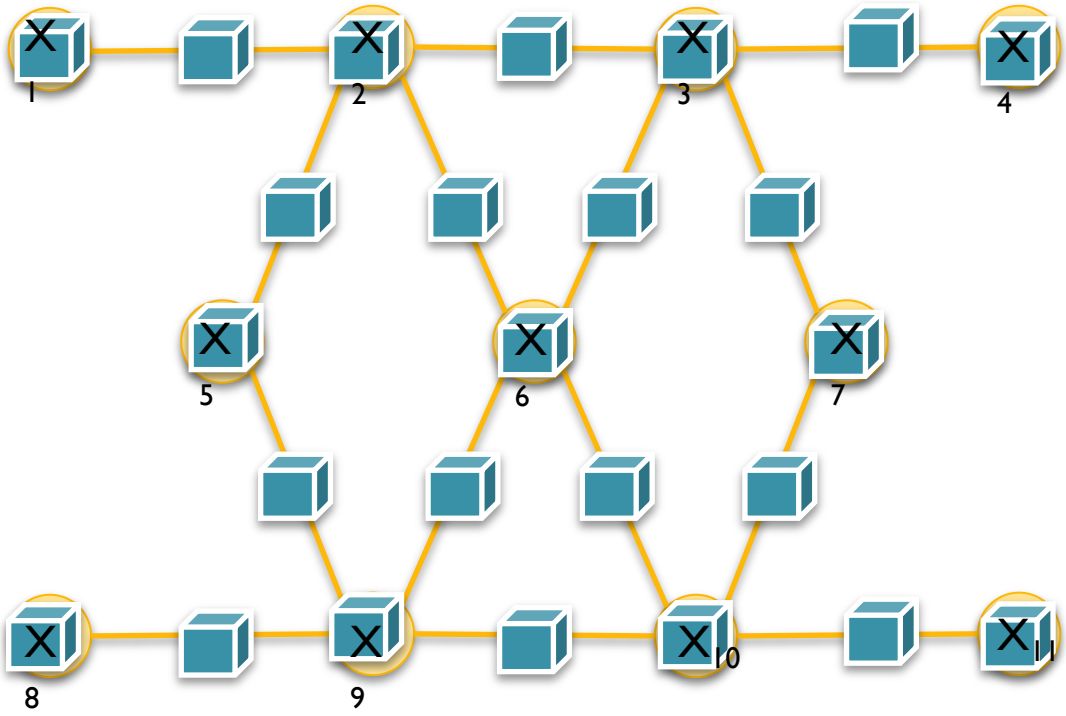
- High-level Abstract
  - Express data dependencies
  - Iterative
- Automatic *Multicore* Parallelism
  - Data Synchronization
  - Consistency
  - Scheduling

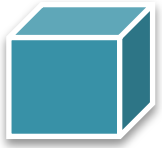
Main Components of GraphLab



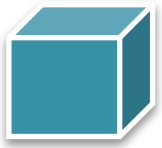
Data Graph

- A Graph with data associated with every **vertex** and **edge**.





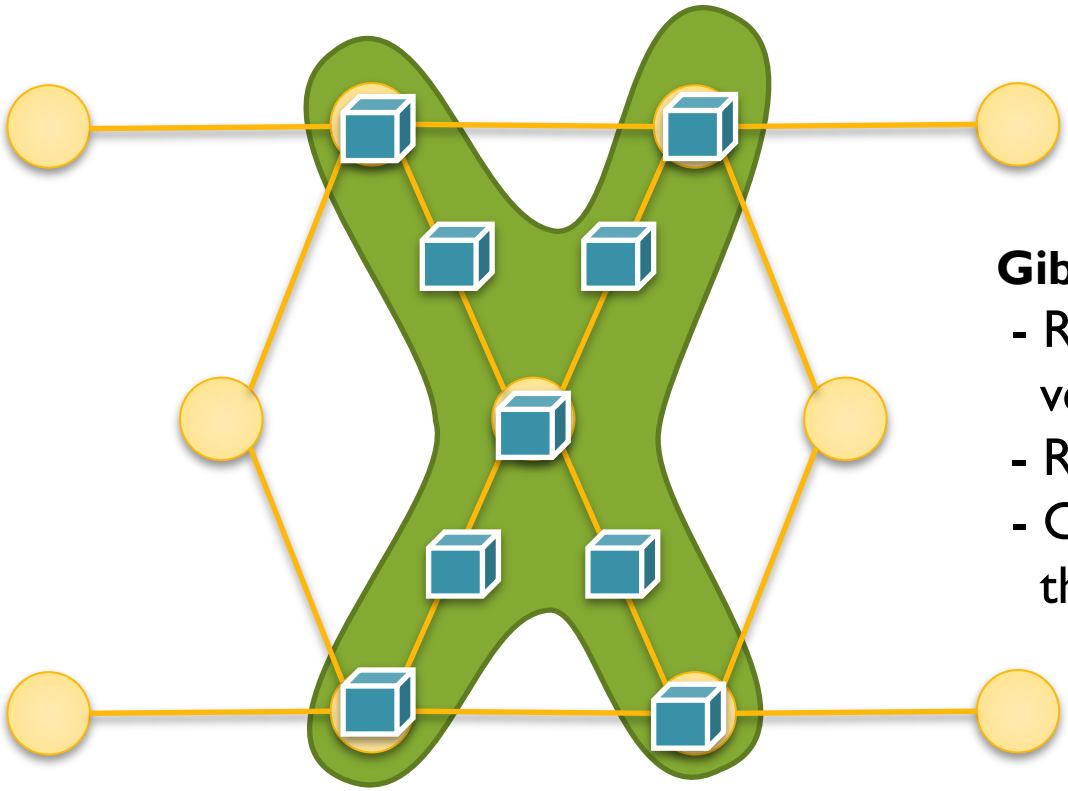
x_3: Sample value
$C(X_3)$: sample counts



$\Phi(X_6,X_9)$: Binary potential

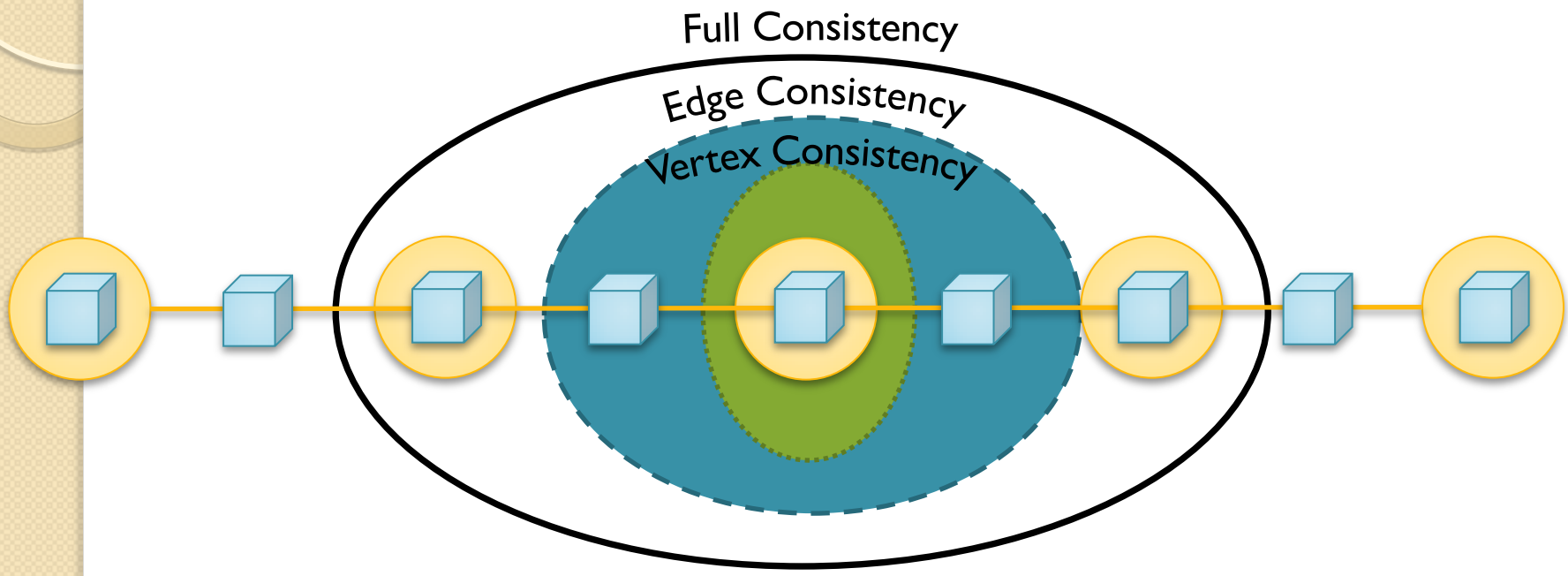
Update Functions

- Operations applied on a **vertex** that transform data in the **scope** of the vertex



Gibbs Update:
- Read samples on adjacent vertices
- Read edge potentials
- Compute a new sample for the current vertex

Scope Rules

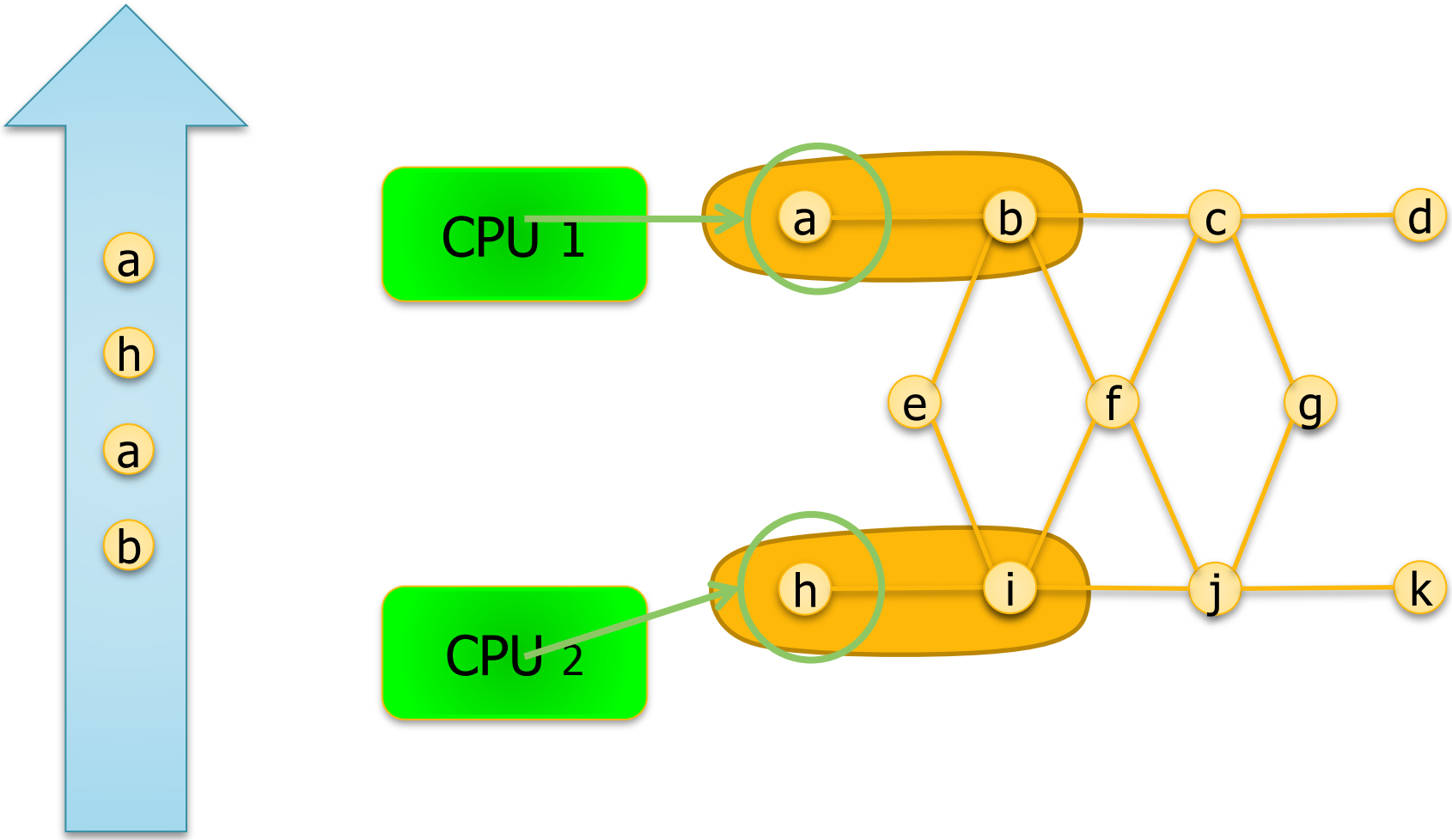


- Consistency v.s. Parallelism
  - Belief Propagation: Only uses edge data
  - Gibbs Sampling: Needs to read adjacent vertices

Scheduling

- Scheduler determines the order of Update Function evaluations
- Static Scheduling
  - Round Robin, etc
- Dynamic Scheduling
  - FIFO, Priority Queue, etc

Dynamic Scheduling



Global Information

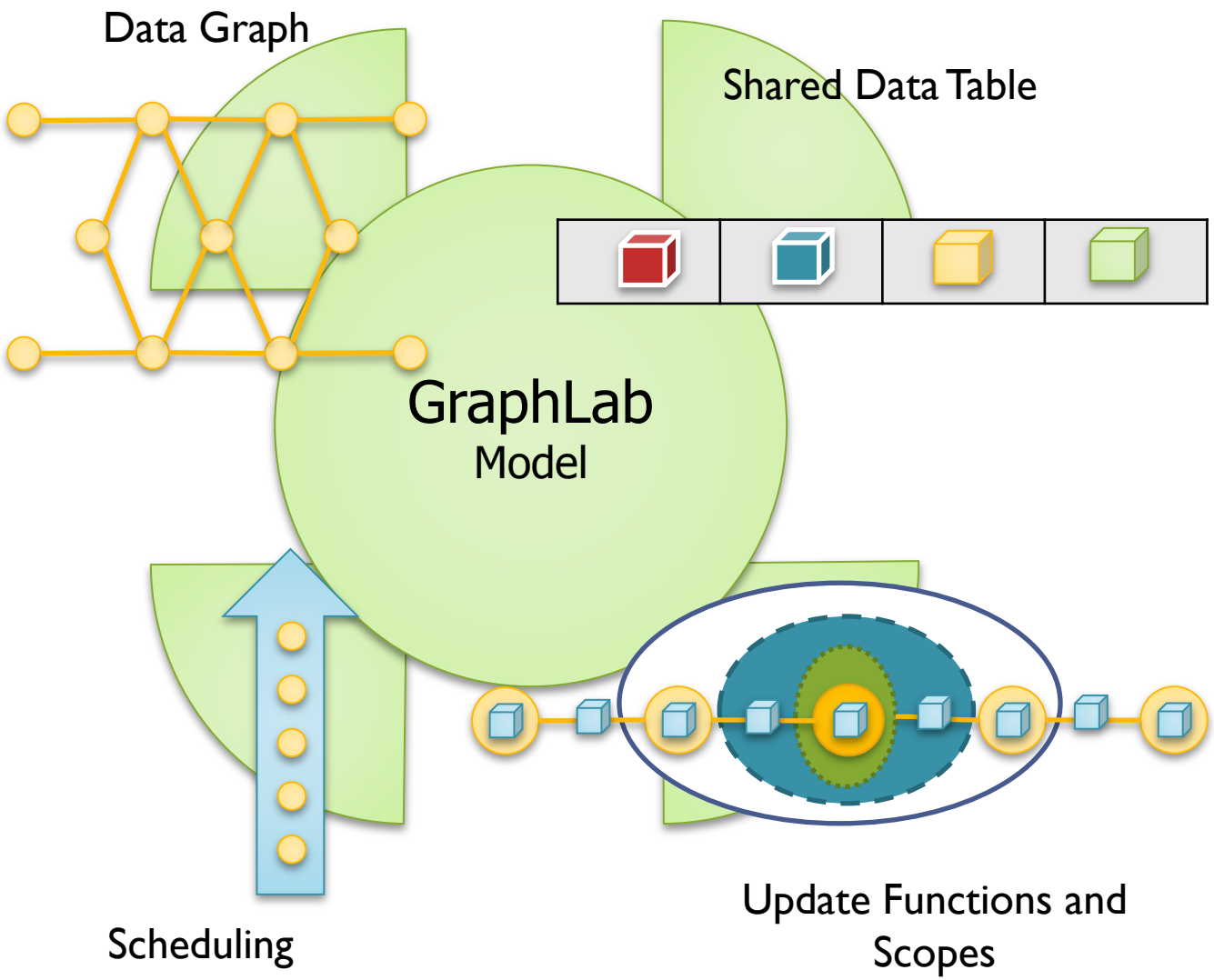
- Shared Data Table in Shared Memory
  - Model parameters (updatable)
  - Sufficient statistics (updatable)
  - Constants, etc (fixed)

- Sync Functions for Updatable Shared Data
  - ***Accumulate*** performs an aggregation over vertices
  - ***Apply*** makes a final modification to the accumulated data

Sync Functions

- Much like Fold/Reduce
  - Execute *Aggregate* over every vertices in turn
  - Execute *Apply* once at the end
- Can be called
  - Periodically when update functions are active (asynchronous) or
  - By the update function or user code (synchronous)

GraphLab


Data Graph
Shared Data Table
GraphLab
Model
Scheduling
Update Functions and
Scopes

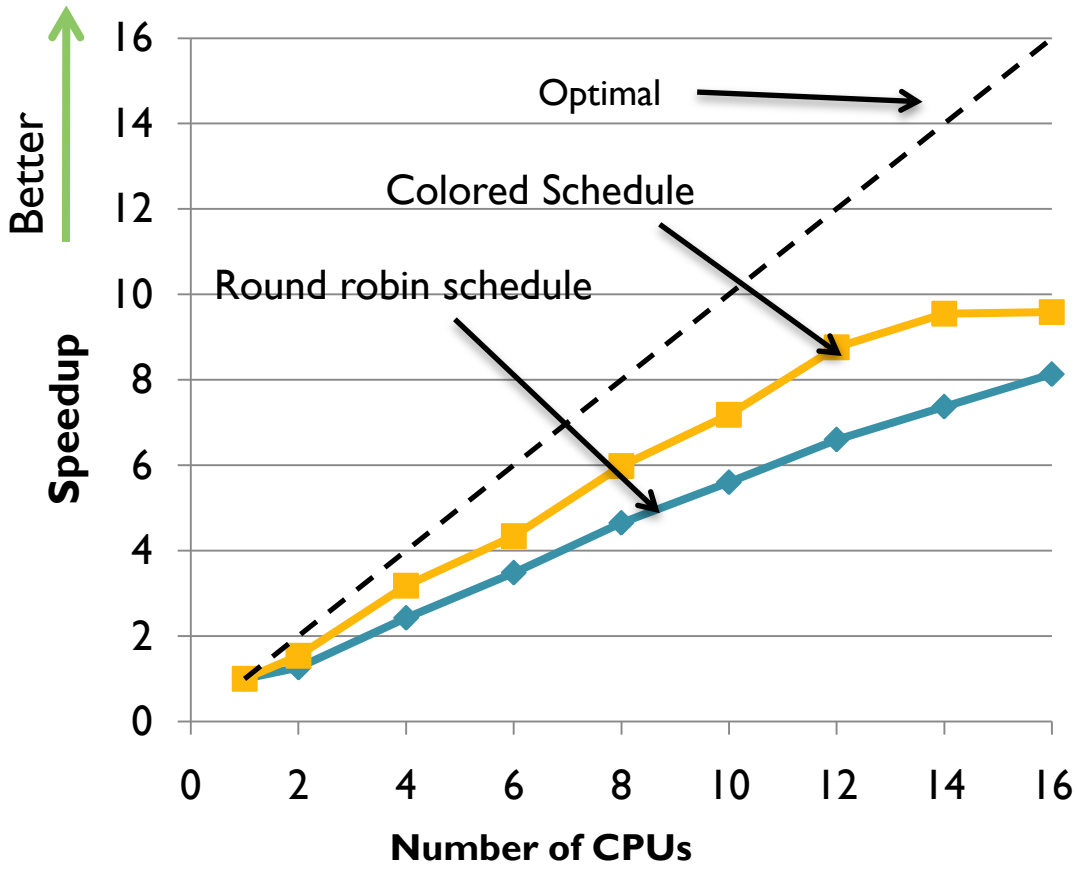
Overview

- Programming ML Algorithms in Parallel
  - Common Parallelism and MapReduce
  - Global Synchronization Barriers
- GraphLab
  - Data Dependency as a Graph
  - Synchronization as Fold/Reduce
- Implementation and Experiments
- From Multicore to Distributed Environment

Implementation and Experiments

- Shared Memory Implemention in C++ using Pthreads
- Applications:
  - Belief Propagation
  - Gibbs Sampling
  - CoEM
  - Lasso
  - etc (more on the project page)

Parallel Performance



From Multicore to Distributed Enviroment

- MapReduce and GraphLab work well for Multicores
  - Simple High-level Abstract
  - Local computation + global synchronization

- When Migrate to Clusters
  - Rethink ***Scope*** → synchronization
  - Rethink ***Shared Data*** → single "reducer"
  - Think ***Load Balancing***
  - Maybe think ***abstract model***?

Thanks