Scalable I/O-Bound Parallel Incremental Gradient Descent for Big Data Analytics in GLADE

Chengjie Qin and **Florin Rusu**
University of California, Merced

June 23, 2013

Motivation

Bismarck [Feng, Kumar, Recht, and Re: Towards a Unified Architecture for in-RDBMS Analytics, SIGMOD 2012]

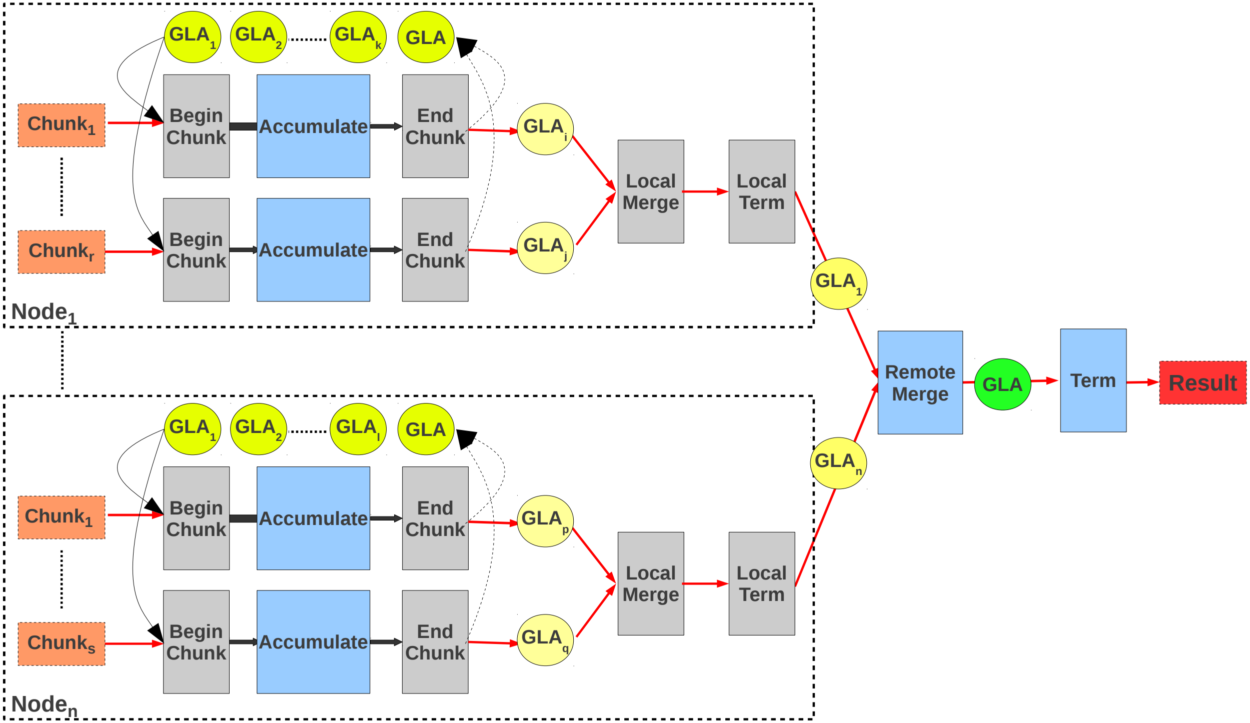
- User-Defined Aggregates (UDA) as general mechanism to represent analytics
- Shared memory
- **Scalability?**

Task	Dataset			System			
	Name	# examples	Size	Bismarck PostgreSQL	DBMS A	DBMS B	In-memory tools
LR	Classify300M	300M	135GB	Yes	Yes	Yes	No
SVM				Yes	Yes	No	No
LMF	Matrix5B	5B	100GB	Yes	N/A	No	No
CRF	DBLP	2.3M	7.2GB	Yes	N/A	N/A	No

Table 1: [Feng, Kumar, Recht, and Re: *Towards a Unified Architecture for in-RDBMS Analytics*, SIGMOD 2012]

GLADE

- k-means clustering (iterative processing): 1-node vs PostgreSQL: 4s vs 62s (**15.5X**); 9-node vs Hadoop: 4s vs 1,297s (**324.25X**)



Research Questions

- What does it take to implement parallel incremental gradient descent (IGD) in GLADE?
- What is the performance of IGD in GLADE?
- Why is GLADE relevant for the Cloud?

Task	**Dataset**			**System**	
	Name	**# examples**	**Size**	**Bismarck PostgreSQL**	**GLADE**
LR	Classify300M	300M	135GB	Yes	?
SVM				Yes	?
LMF	Matrix10B	10B	200GB	Yes?	?
CRF	DBLP	2.3M	7.2GB	Yes	?

Agenda

- Predictive analytics
- Gradient descent
- Incremental (stochastic) gradient descent
- Parallel IGD
- GLADE
- Parallel IGD in GLADE
  - IGD as GLA
  - Iteration management
  - Randomization
  - Merging
- Answer to research questions

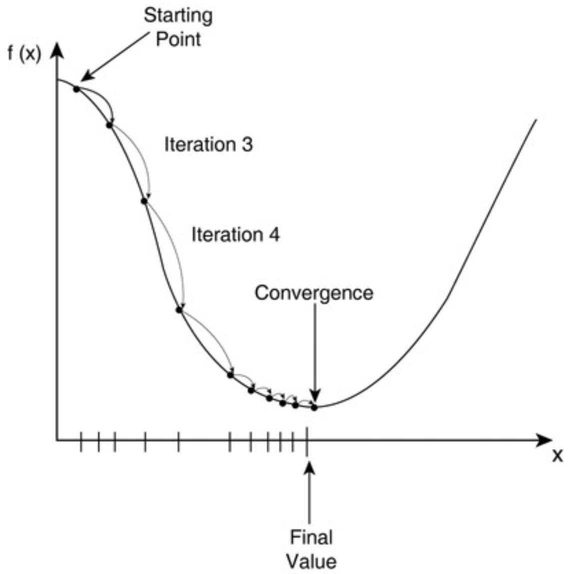
Predictive Analytics

- Massive amounts of example data, e.g., 10 billion
- Data are highly-dimensional, e.g., 50 to 600 million

Analytics task	Objective function
Logistic Regression (LR)	$\sum_{(x_i,y_i)\in\text{data}} \log\left(1+e^{-y_i w^T x_i}\right)+\mu\|\vec{w}\|_1$
Classification (Support Vector Machines - SVM)	$\sum_{(x_i,y_i)\in\text{data}} \left(1-y_i w^T x_i\right)+\mu\|\vec{w}\|_1$
Recommendation (Low-Rank Matrix Factorization - LMF)	$\sum_{(i,j)\in\Omega} \left(L_i^T R_j - M_{ij}\right)^2+\mu\|L,R\|_F^2$
Labeling (Conditional Random Fields - CRF)	$\sum_{(x_i,y_i)\in\text{data}} \left[\sum_j w_j F_j\left(y_i,x_i\right)-\log Z\left(x_i\right)\right]$

Table 2: [Feng, Kumar, Recht, and Re: *Towards a Unified Architecture for in-RDBMS Analytics*, SIGMOD 2012]

Gradient Descent



http://www.yaldex.com/game-development/1592730043_ch18lev1sec4.html

$$\min_{w \in \mathbb{R}^d} \sum_{(x_i, y_i) \in \text{data}} f(w, x_i, y_i)$$

$$w^{(k+1)} = w^{(k)} - \alpha_k \nabla f\left(w^{(k)}\right)$$

∇f is the gradient

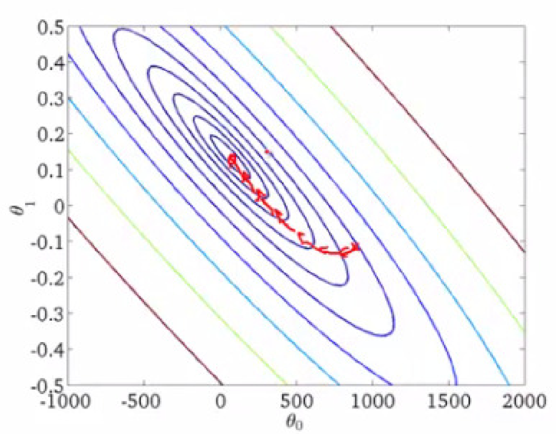
α_k is step size or learning rate

$w^{(0)}$ is the starting point (random)

- Single step is taken for every iteration over data: slow convergence
- Step size is data-dependent: ping-pong effect; line search is expensive
- Convergence to minimum guaranteed for convex objective function

Incremental Gradient Descent (IGD)

Gradient descent

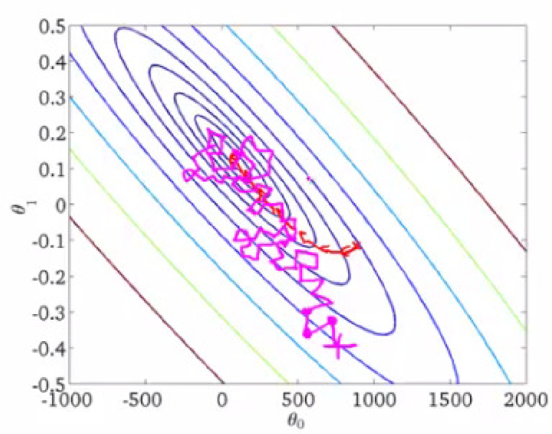


http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html

$$w^{(k+1)} = w^{(k)} - \alpha_k \nabla f\left(w^{(k)}\right)$$

- Exact gradient computation
- Single step for one iteration
- Faster convergence close to minimum

Incremental gradient descent



http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html

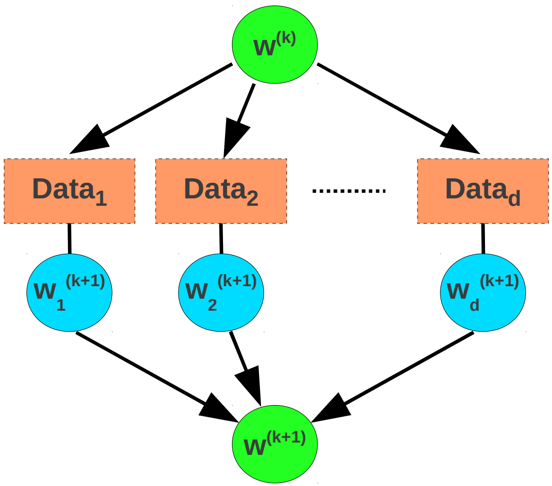
$$w^{(k+1)} = w^{(k)} - \beta_k \nabla f_{\eta(k)}\left(w^{(k)}\right)$$

- Approximate gradient at data point
- Take step for each data point
- Faster convergence far from minimum

Parallel IGD

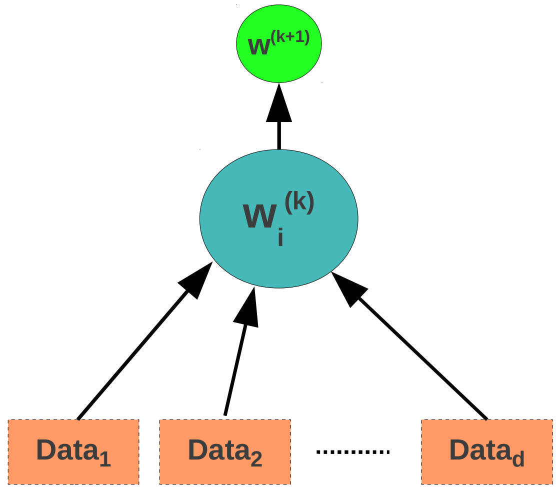
- Assumption: IGD is commutative and algebraic

Distributed model



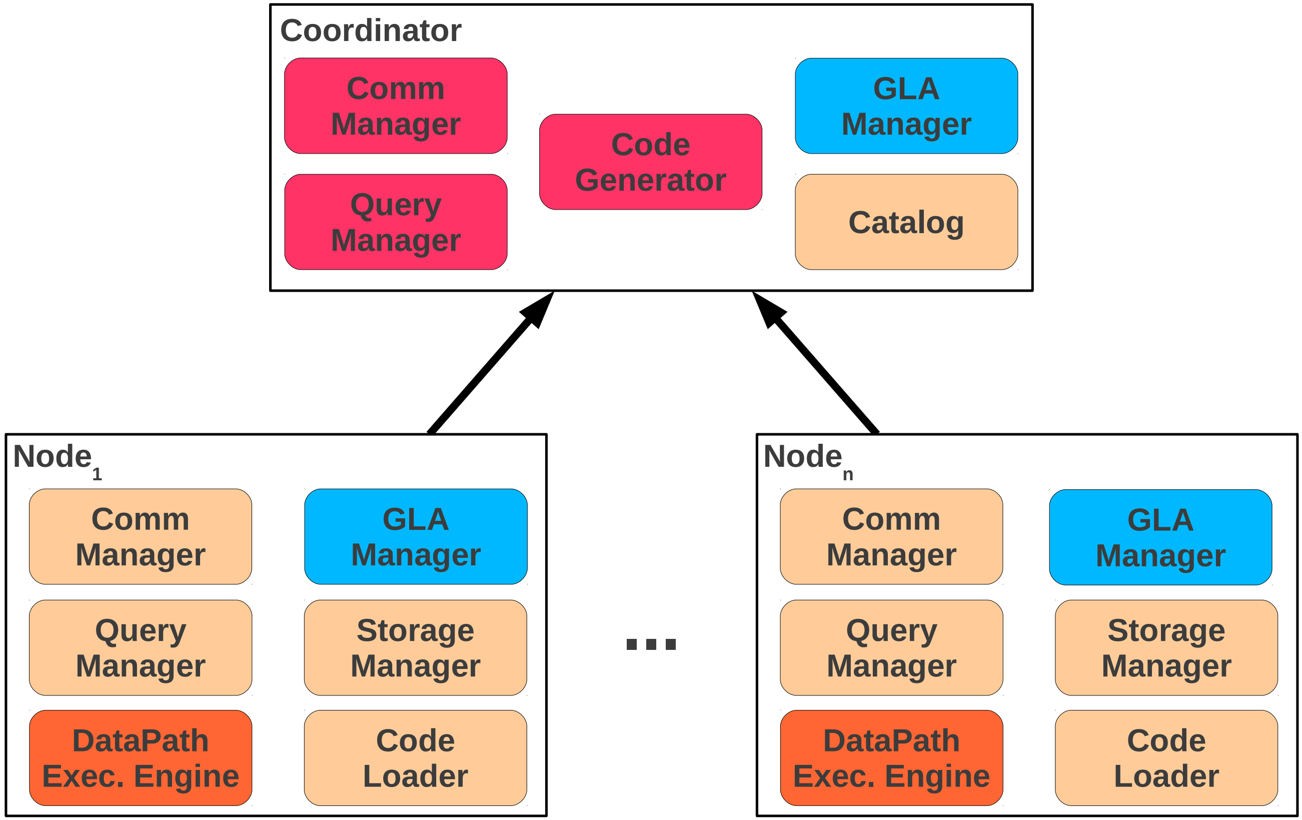
- Distribute model
- Merge partial models
- Scalable

Shared model

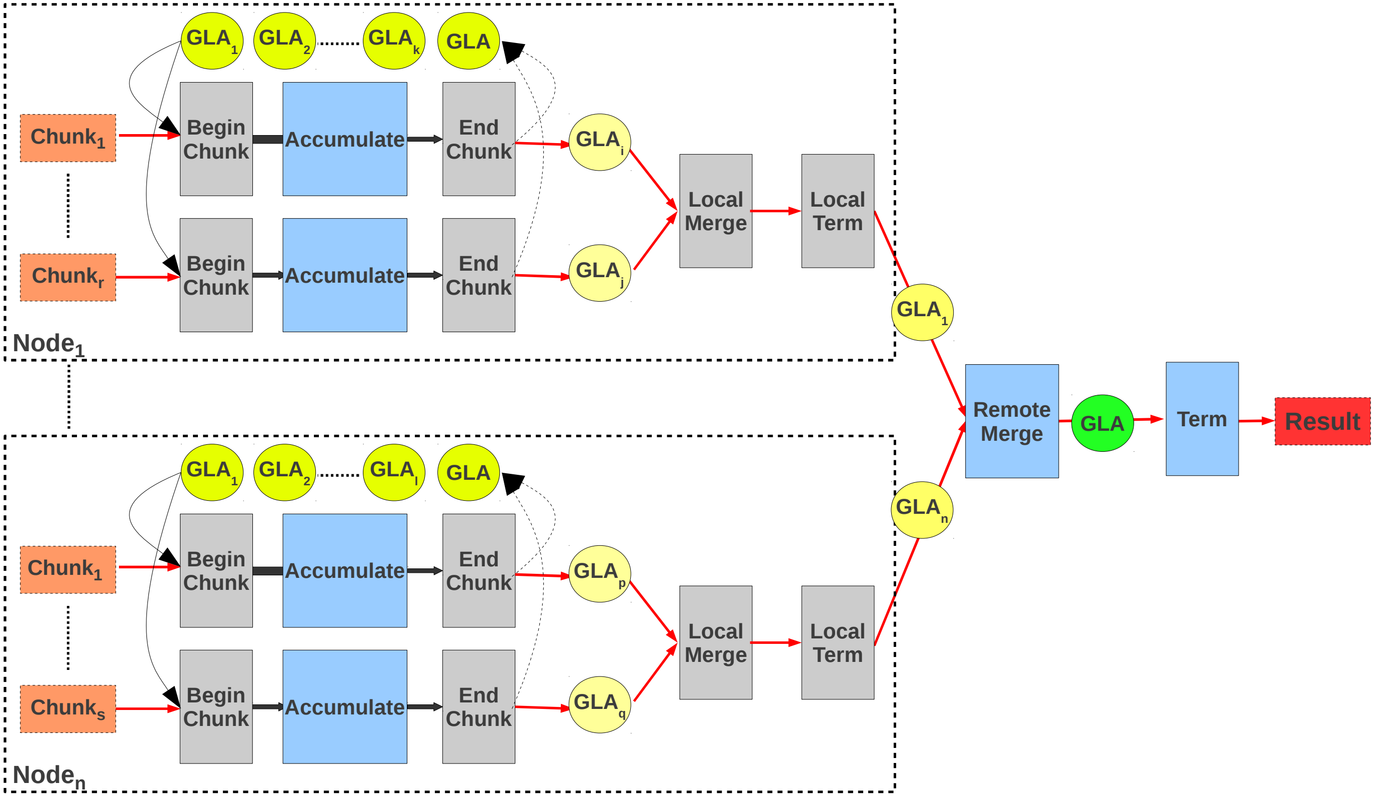


- Synchronize access to model
- Limited scalability

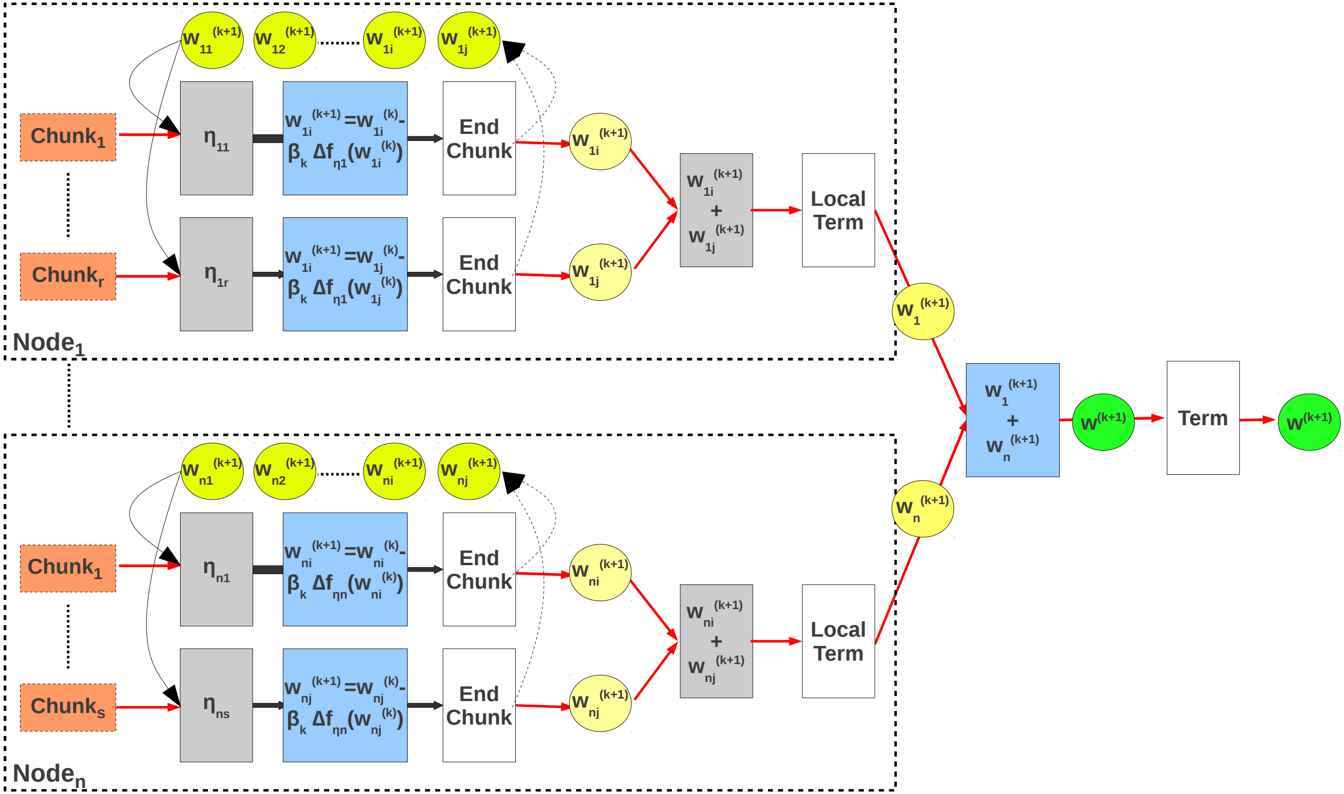
GLADE Architecture



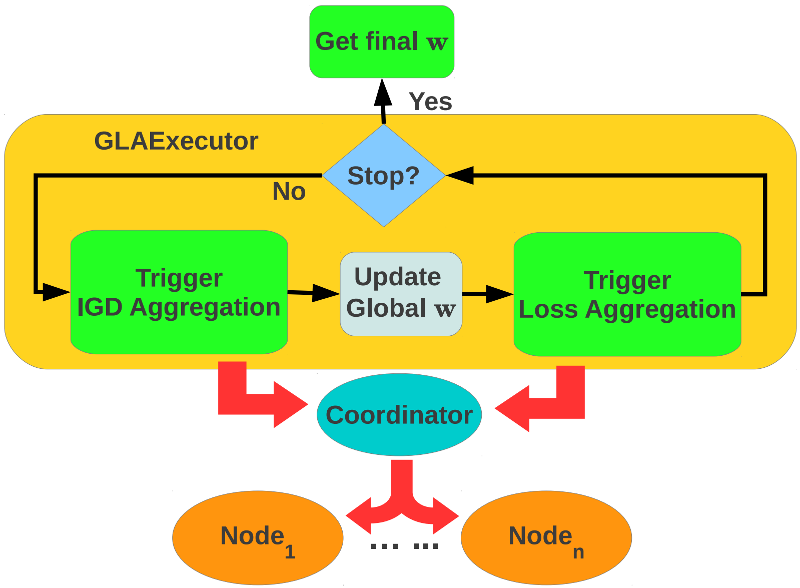
GLADE Execution Model



IGD as Generalized Linear Aggregate (GLA)



Iteration Management

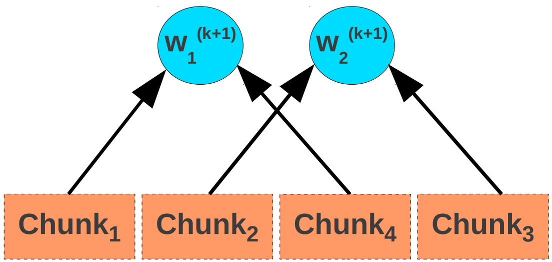


- Stop condition
  - Compute error on consecutive models; fix number of iterations
- Pass model between iterations
  - File system (NFS); broadcast message

Data Randomization

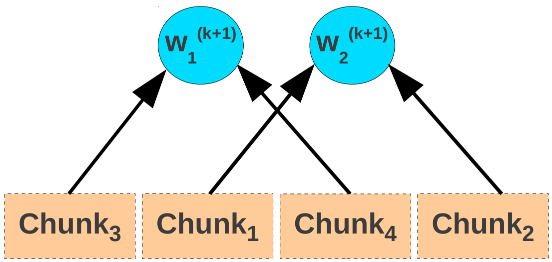
- Randomize data during initial loading process: partition on random hash function

Implicit randomization

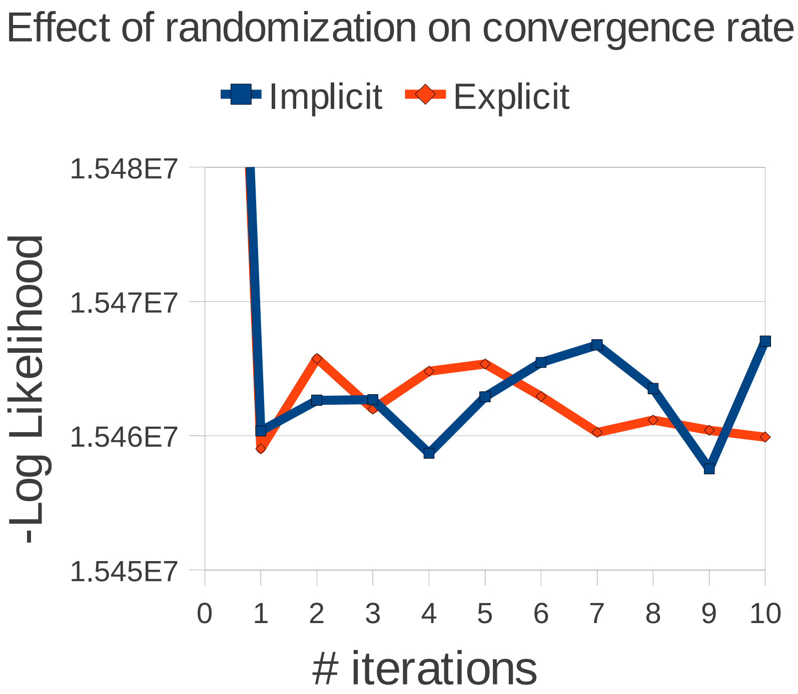


- Chunks are dropped non-deterministically
- Assignment of chunks to GLAs is non-deterministic
- No changes to GLADE

Explicit randomization

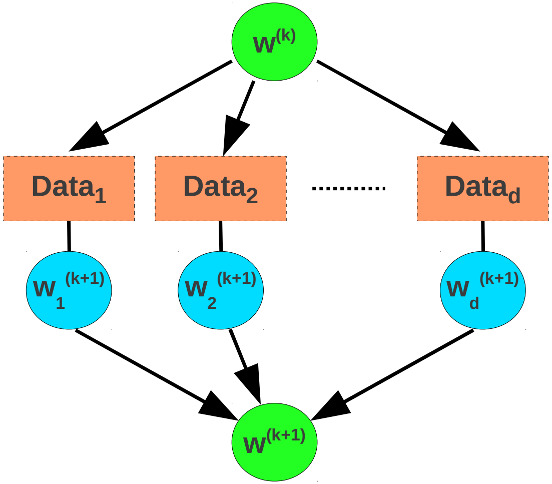


- Randomize scan order of chunks from disk: `StorageManager`
- Randomize scan order of tuples inside chunk: `BeginChunk`

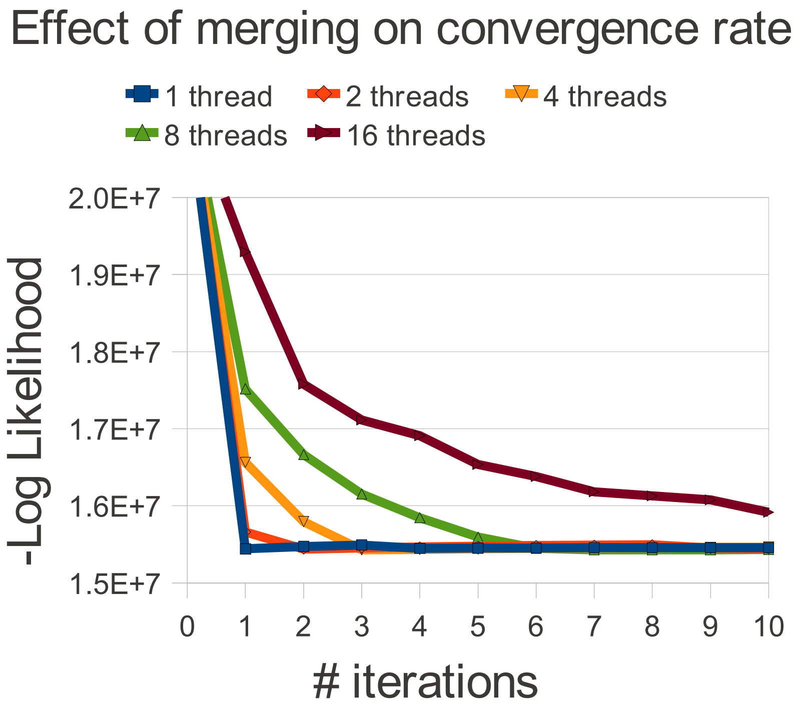


- LR on `Classify300M`
- Single node, single thread

Model Merging

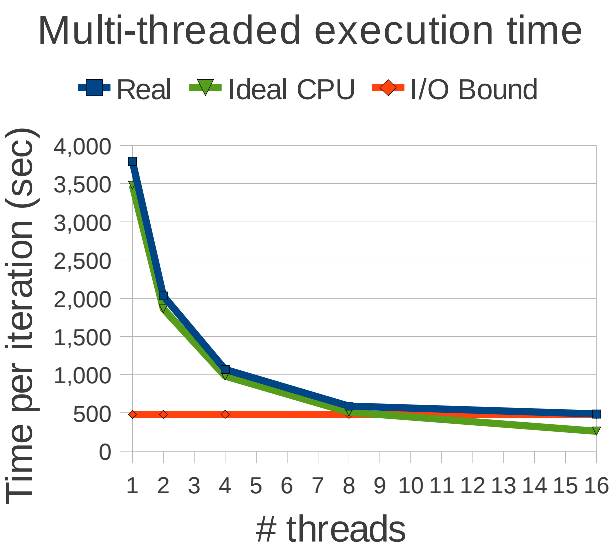


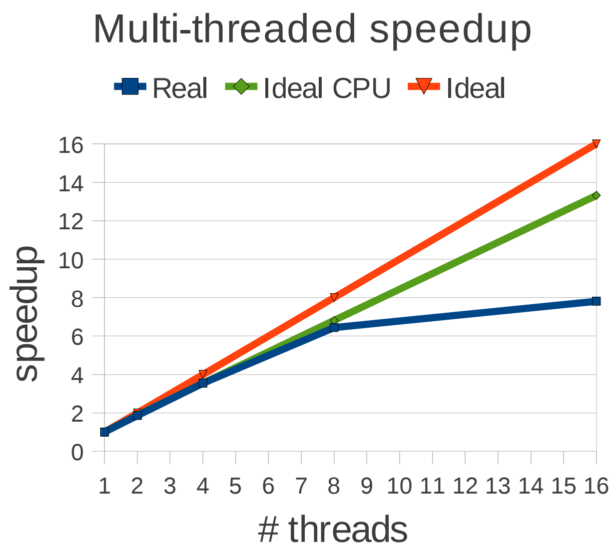
- Strategies
  - (Weighted) average models component-wise
  - Ensembles
- `LocalMerge` and `RemoteMerge` provide support for any merging strategy, including hybrid



- LR on `Classify300M`
- Single node

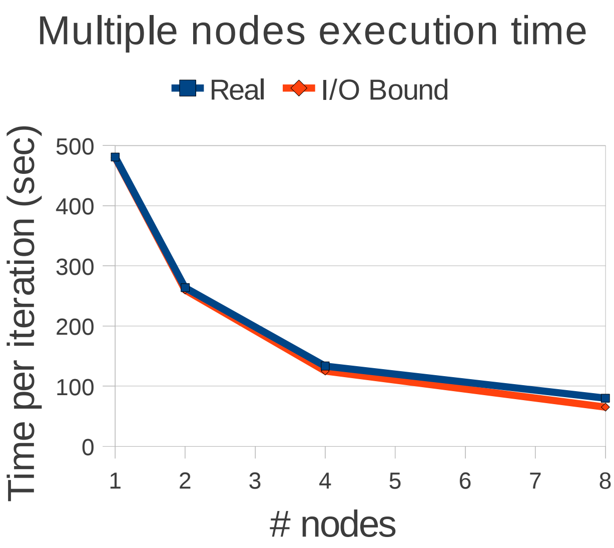
Single Node Performance

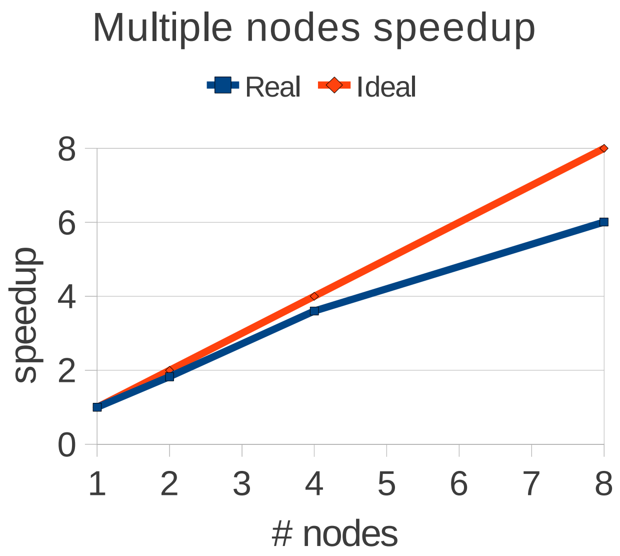




- LMF on `Matrix10B`
- Single node: 16 cores @ 2GHz; 16GB RAM; 4 disks @ 110MB/s throughput/disk

Cluster Performance





- LMF on `Matrix10B`
- Single node: 16 cores @ 2GHz; 16GB RAM; 4 disks @ 110MB/s throughput/disk
- Cluster: 8 X worker + coordinator (9 nodes); Gigabit Ethernet; same rack

Answer to Research Questions

- What does it take to implement parallel IGD in GLADE?
  - Bismarck UDA-based implementation with conversion in data representation
- What is the performance of IGD in GLADE?
  - I/O-bound
  - Linear scalability on single-node (multi-threaded) and cluster
- Why is GLADE relevant for the Cloud?
  - Full resource utilization for your money
  - **324.25X** on $30,000 9-node cluster vs Hadoop

Task	Dataset			System	
	Name	# examples	Size	Bismarck PostgreSQL	GLADE
LR	Classify300M	300M	135GB	Yes	**31.94 sec/iteration**
SVM				Yes	**32.15 sec/iteration**
LMF	Matrix10B	10B	200GB	Yes?	**80.04 sec/iteration**
CRF	DBLP	2.3M	7.2GB	Yes	**969.23 sec/iteration**

Questions