Motivation

- Growing number of extremely large datasets
  - Terabytes/day of search user interaction data
- Cheaper hardware
- Decreased model training time allows for solutions to more problems (faster throughput)
- **Distribute LambdaMART** [Wu et al., 2009]
  - Gradient boosted decision trees, where gradients are approximated using $\lambda$-functions
  - Constructed incrementally $f(x, N) = \sum_{n=1}^{N} h_n(x)$
    - $N$: # boosting iterations; $x$: sample feature vector; $h_n(x)$: weak hypothesis
- Key component of winning system in Yahoo! Learning to Rank Challenge
Our Approach

**Feature distribution**
- Data fits in main memory of a single machine
- Tree split computations are distributed
- Learned model equivalent to centralized model

**Data distribution**
- Data does not fit in main memory of a single machine
- Data is distributed and used for cross-validation
- Learned model not equivalent to centralized model
- Two strategies for selection of next weak hypothesis (full and sample)
Two types of cluster nodes.
- Features split across nodes.
- Each node stores all of the training data.
- Centralized (dotted)
- Feature-distributed (solid)

Feature distribution yields 6 times speed-up
- Data split across nodes.
- Each node stores 3500 queries. (Centralized stores 3500*K queries).
- Centralized (dotted)
- Full data-distributed (solid)
- Sample data-distributed (dashed)

Data distribution yields 10 times speed-up
Data distribution yields significant accuracy gain when data size exceeds main memory size.

- Data split across nodes.
- Each node stores 3500 queries.
- Full data-distributed (solid)
- Sample data-distributed (dashed)
Centralized model significantly better than data-distributed model. More training data better than massive cross-validation.

- Data split across nodes.
- Each node stores 3500 queries. (Centralized stores 3500*K queries).
- Centralized (dotted)
- Full data-distributed (solid)
- Sample data-distributed (dashed)
Future Work

Performing massive, distributed cross-validation results in significant accuracy loss compared to learning from centralized training data.

Can we distribute data and achieve equal/better accuracy while decreasing training time?

Can we develop an asynchronous approach resulting in additional speed improvements?

Thanks!