1. The Problem

Highly-distributed data and computation:
- CDN log data at 1000s of locations
- Global weather sensors
- Internet business with several data centers

Original MapReduce assumptions broken:
- Data & compute resources may not all be co-located.
- Data may not exist within compute cluster.
- Interaction between phases becomes a much greater challenge.

Objective: Improve MapReduce performance when these assumptions no longer hold.

2. Current Approaches

Centralized computation (push to central location, then compute):
- Requires sufficient centralized resources
- Imbalanced: Some links and compute nodes over-utilized, others idle
  (but could be OK if total data small, needs to be archived, or reused in many computations)

Simple dynamic load balancing:
- Speculative execution & lazy task assignment (i.e., work-stealing)
- Myopic: Fail to consider interaction between phases

3. New Approach: Model and Optimize

Model:
- Network-aware: link speeds
- Application-aware: compute speeds, rate of data expansion in map, α
- End-to-end: model how phases fit together, interact

Optimization:
- End-to-end: optimize entire makespan rather than...
- Myopic: optimize push, then optimize shuffle

Implementation, Validation:
- Prototype in Hadoop 1.0.1
- Tightly couple data and task placement
- Control data/task placement in Map, Reduce

4. End-to-End Outperforms Myopic

Result: Although a myopic approach is better than nothing, end-to-end optimization is much more effective.

5. End-to-End Optimization Outperforms Hadoop

Result: End-to-end optimization may sacrifice phase-level optimality to achieve optimal end-to-end makespan. Optimized execution plans outperform Hadoop.

6. Ongoing Work

Existing dynamic mechanisms:
- Tend to help Hadoop
- Tend to undermine optimized execution plan

Challenges:
- Make end-to-end optimization more adaptable
- Make dynamic mechanisms smarter