Managed Communication for Fast, Large-Scale, Iterative Analytics
Jinliang Wei, Wei Dai, Henggang Cui, Qirong Ho, Greg Ganger, Phil Gibbons*, Garth Gibson, Eric Xing (CMU, *Intel Labs)

Background & Motivation
• Inter-machine communication is the bottleneck when scaling out
  • Parameter server & client stubs manage communication
• Application tolerance of bounded staleness allows delayed communication (Stale Synchronous Parallel)
  • Batching & coalescing reduce total communication
  • Communication becomes more bursty and efficient
  • But reduced delay (freshness) improves convergence progress per update

Evaluation
• Hardware: 8 x 64-core machines with 1GE
• Application: Topic Modeling
  • Specifically, LDA with Gibbs sampling [1]
  • Using NYTimes dataset, deriving 1000 topics
  • Parameters and data fit in memory
• Parameters (\(a_i\)) organized in rows, app requires atomic batch (row) updates
• Importance of parameter update (\(u_i\)) becomes importance of row update
  • E.g. per row, \(\text{max}(|u_i|)\) or \(\text{sum}(|u_i/a_i|)\)
    (we use the latter)
• Compare no early update, random row early update and “largest” row early update

Absolute Convergence Rate Improves with Early Update

Managed Bandwidth Stale Synchronous Parallel (MBSSP)
• Communicate most important updates early and with bounded inter-machine bandwidth
  • Efficiency from potentially large delay, and convergence benefits of freshness where it counts
• Application-specific policies for prioritizing updates to communicate early
  • Eg., absolute magnitude of parameter delta
  • Eg., relative magnitude of delta over parameter

Convergence per sec
-1.0
-1.1
-1.2
-1.3
-1.4
-1.5
50 100 150 200 250 300 350 400 450 500 seconds
MBSSP-Random, 600Mbps
MBSSP-Largeness, 400Mbps
MBSSP-Largeness, 800Mbps
SOP slack = 2

Bandwidth Usage

• 800 Mbps max bandwidth provides 40% speedup with random prioritization
• While prioritizing by row largeness achieves better speedup with 400 Mbps max

Conclusion & Future Work

Our vision:
• Updates bandwidth should be carefully managed
• Prioritization of “important” updates is app specific

Future work:
• Application-specific rules for suppressing updates
• Further delaying, dropping, lossy compression…
• Semantics for application-specific consistency requirements