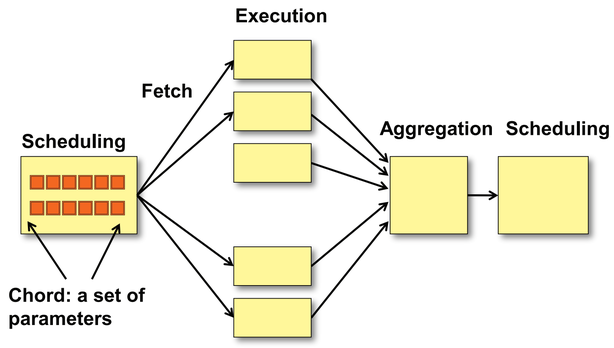
STRADS: Parallel ML Scheduling for High Efficiency

Jin Kyu Kim, Seunghak Lee, Eric Xing, Garth Gibson (CMU)

ML Iterative Solver Execution Model

- Scheduling/Fetch/Execution/Aggregation model

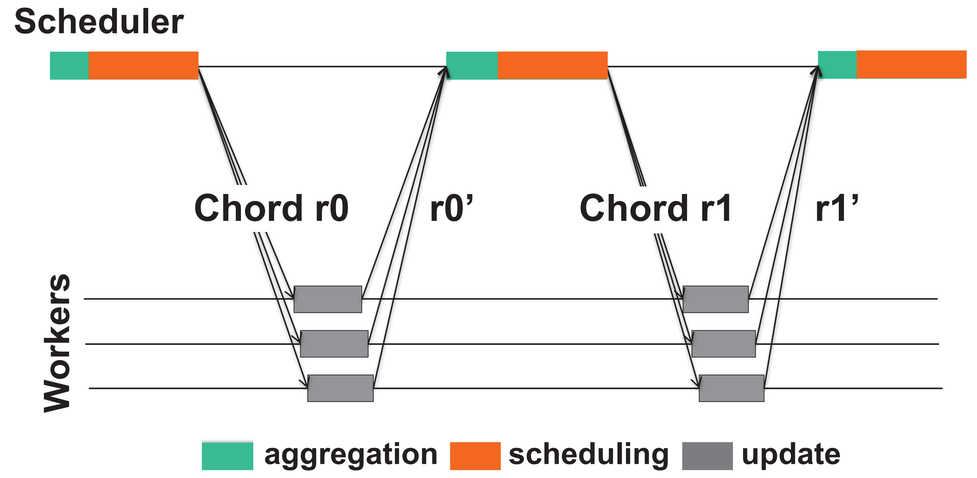


- Some ML apps are intolerant of massive parallelism (Ex. Lasso)
- STRADS selects chord to minimize aggregate errors of parallel update

→Parameters of a chord are approximately independent

Example Timeline

- Serial execution of chords is a performance bottleneck

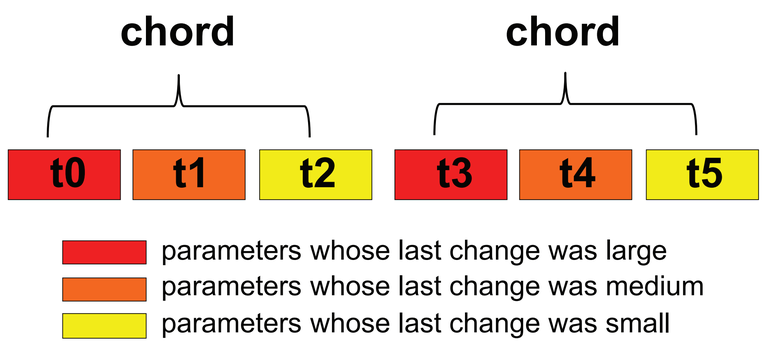


- Issue: How do we keep all worker cores busy and effective?

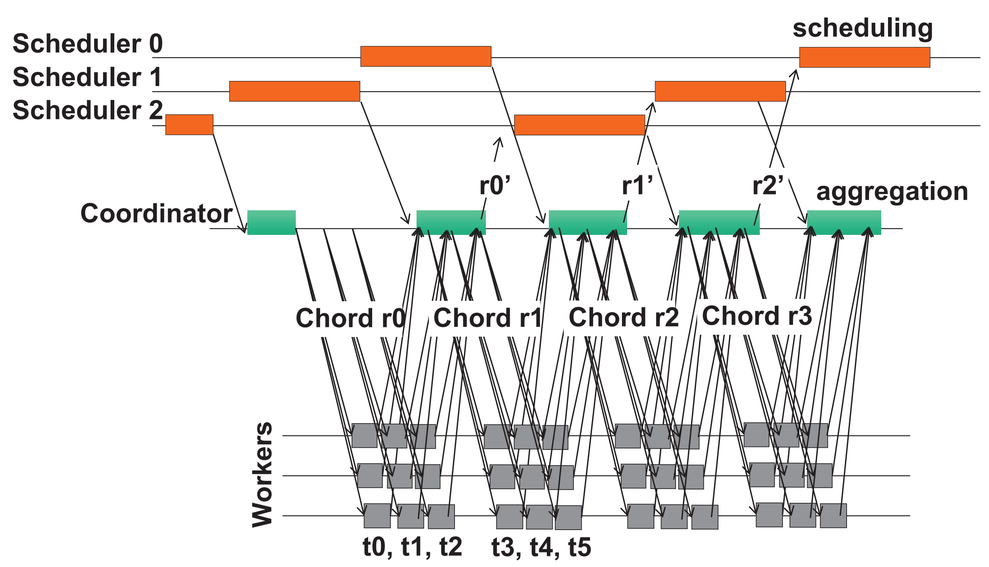
High Throughput Scheduler

1. Pipeline multiple schedulers to hide scheduling latency
   - Divide parameter space into disjoint partitions (one per scheduler)
   - Scheduling decisions depend only on local partition
   - Update execution will see globally fresh data

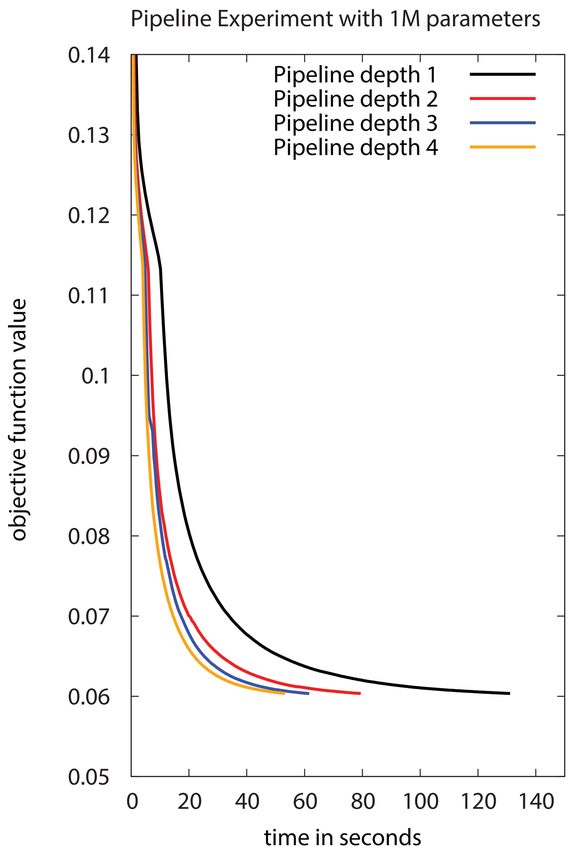
2. Pipeline within one chord to hide communication latency
   - Allow next chord to start execution before all results of current chord are globally known
   - Scheduler prioritizes most impactful (largest) updates to front of pipeline
   - Restrict updates that might not might not be fresh to least impactful updates
   - Ensure most impactful updates of consecutive chords see fresh results (t3 see t0's results)



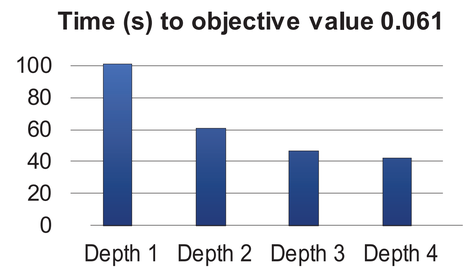
Dual Pipeline Better Utilizes Workers



Pipeline Experiment



- Application: Lasso
- Data: Synthetic data
- 50K samples, 1M dimension



Conclusions

- Dual pipeline better utilizes workers and improves convergence speed
- Three canonical ML applications (Lasso, Logistic Regression, SVM) implemented in STRADS framework so far

Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley · University of Washington