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

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 be fresh to least impactful updates
   - Ensure most impactful updates of consecutive chords see fresh results (t3 see t0’s results)

Conclusions

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

Example Timeline

- Serial execution of chords is a performance bottleneck

Dual Pipeline Better Utilizes Workers

Pipeline Experiment

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