Paper Review - MAST: Global Scheduling of ML Training across Geo-Distributed Datacenters at Hyperscale

Summary: This paper introduces MAST (ML Application Scheduler on Twine), a global scheduling system developed at Meta to address the severe load imbalance and poor hardware utilization caused by users manually selecting datacenter regions for their machine learning (ML) training data and computation. In Meta's private cloud, this manual process led to major mismatches between workload demand and hardware supply, with the most overloaded region showing a GPU demand-to-supply ratio of 2.63 for high-priority workloads. MAST works alongside Meta's Resource Assignment Service (RAS), which groups machines into dynamic ML clusters, and Twine, which manages container orchestration, to coordinate efficient global training execution. MAST provides a unified global-scheduling abstraction that automatically places both data and ML workloads across multiple regions. The system is built on three major design principles: temporal decoupling, which separates real-time job scheduling (fast path) from background data and machine optimization (slow path, using Tetris, a system that optimizes data placement across regions by considering data-access patterns of Spark, Presto, and ML training jobs and uses soft-balance constraints for scarce GPUs while enforcing hard constraints for CPU and storage); scope decoupling, which assigns responsibilities at different scales (Global ML Scheduler (GMS) for global job queue, Regional ML Scheduler (RMS) for regional resource allocation, and Cluster Manager (CM) for cluster orchestration); and exhaustive search, where RMS evaluates all clusters to find the best placement plan for both data and training jobs. Deployed at hyperscale, MAST reduced the GPU imbalance ratio to 0.98, achieved 98% GPU allocation efficiency, and maintained a data non-collocation rate below 0.1%, confirming its scalability and effectiveness.

Evaluation: The paper tackles a highly significant challenge in large-scale ML infrastructure - efficiently utilizing scarce and expensive GPUs across global datacenters. Its deployment at Meta, managing around 100,000 GPUs across tens of regions, shows strong real-world impact, with overload reduced from 2.63 to 0.98 and GPU allocation reaching 98%. The approach is novel, valid and well-structured around three clear design principles. The use of temporal decoupling to separate slow cross-region data movement from real-time scheduling is particularly effective. The application of scope decoupling and exhaustive search at a global scale, along with soft-balance modeling for GPU scarcity in data placement, makes the design novel compared to prior systems like Hydra and Yugong (both use early binding). The evaluation is strong and well-supported by real production data and workload results. About 70% of workloads are first-time jobs, and the system still achieves a high GPU allocation rate of 98%, showing reliability and scalability. The paper also strengthens its results by comparing three versions of its data placement system (V0, V1, and V2) and by using a simulated federated fast-path scheduler, which helps justify the design choices. The manuscript is well-written and clearly explains both challenges and solutions.

Main takeaways

1. MAST's temporal decoupling - splitting fast, real-time job scheduling from slow, background data movement, ensures efficiency despite long data transfer times across regions.
2. MAST shows stable production-scale operation across Meta's global infrastructure, orchestrating tens of thousands of ML jobs daily with no reported regressions in reliability.

Strengths

1. The design based on temporal decoupling, scope decoupling, and exhaustive search makes complex geo-distributed scheduling understandable and modular.
2. Assigning scheduling responsibilities across global (GMS), regional (RMS), and cluster (CM) levels prevents bottlenecks while maintaining scalability and control.

Weaknesses

1. The evaluation of the Tetris solver is not very detailed, comparisons with commercial MIP solvers or reinforcement-learning-based optimizers could strengthen confidence in its optimality.
2. The reliance on large-scale data replication increases storage usage by 75-125%, which may become unsustainable as datasets and model checkpoints continue to grow.

Discussion: Could MAST be adapted for smaller organizations or public cloud platforms that lack Meta's massive infrastructure and custom systems like Tetris, Twine and RAS?