SHOCKWAVE: FAIR AND EFFICIENT GPU SHARING FOR DEEP LEARNING UNDER DYNAMIC ADAPTATION

Pengfei Zheng, Rui Pan, Tarannum Khan,
Shivaram Venkataraman, Aditya Akella
SHARED CLUSTERS
FOR MACHINE LEARNING
GPU CLUSTER SCHEDULING

Multiplex access to shared GPU cluster for contending DL apps

GPU Cluster Scheduler

Shared GPU cluster
Gradient Noise Scaling (GNS)

Adaptively double batch size based on gradient noise
Gradient Noise Scaling (GNS)

Adaptively double batch size based on gradient noise

Small Batch Size (16),
Large Batch Size (4096)
Batch size 16 $\rightarrow$ 32 $\rightarrow$ ... $\rightarrow$ 4096

KungFu (OSDI 2020)
DYNAMIC ADAPTATION IN ML JOBS: ACCORDION

Use small batch size in critical regimes, large batch size elsewhere
Accordion (MLSys 2021)
Dynamic adaptation can affect Efficiency
Dynamic adaptation can affect Efficiency
SCHEDULING WITH DYNAMIC ADAPTATION

Dynamic adaptation can affect Efficiency
SCHEDULING WITH DYNAMIC ADAPTATION

Dynamic adaptation can affect **Efficiency**
Dynamic adaptation can affect **Fairness**

- Leads to 2x worse than fair finish time
Dynamic adaptation can affect **Fairness**

Leads to 2x worse than fair finish time

How do we achieve efficiency with long term fairness for dynamic ML jobs?
Shockwave: Fair and Efficient Scheduler under Dynamic Adaptation

Key contributions

- **Predictive model** for Dynamic Adaptation
- **Market-based** formulation with time-varying utility
- Improves makespan by ~1.3x and fairness by ~2x
OUTLINE

Problem statement
Related work, Challenges
Shockwave Design
Evaluation
SOME PRIOR DL SCHEDULERS

Improve Cluster Utilization:
Gandiva (OSDI 2018), AntMan (OSDI 2020), HiveD (OSDI 2020)

Reduce Job-completion Time
Tiresias (NSDI 2019), Optimus (Eurosys 2019)

Optimize Goodput through Elasticity
SLAQ (SoCC 2017), Optimus (Eurosys 2019), Pollux (OSDI 2021)

Fair sharing across users:
Themis (NSDI 2020), Gandiva_{fair}, AlloX (Eurosys 2020), Gavel (OSDI 2020)
CHALLENGE: INACCURATE ESTIMATES

Themis Objective: \( \min (\max \rho) \)

Interface: Get finish time fairness \( (\rho) \) estimates from all apps

\[
\rho_1 > \rho_2 > \rho_3 > \ldots > \rho_N
\]
**CHALLENGE:** INACCURATE ESTIMATES

**Themis Objective:** \( \min (\max \rho) \)

**Interface:** Get finish time fairness (\( \rho \)) estimates from all apps

\[
\rho_1 > \rho_2 > \rho_3 > \ldots > \rho_N
\]
CHALLENGE: FILTERING ON PAST ALLOCATIONS

Themis objective – min (max \( \rho \))

\[
\rho_1 > \rho_2 > \rho_3 > \cdots > \rho_N
\]

\[
l-f \quad \quad f
\]

1. Filter \( l - f \) apps with worst \( \rho \) values
2. Allocate to one or more of \( l - f \) apps for next round using partial auctions
CHALLENGE: FILTERING ON PAST ALLOCATIONS

Themis objective – min (max $\rho$)

1. Filter $l - f$ apps with worst $\rho$ values
2. Allocate to one or more of $l - f$ apps for next round using partial auctions
SHOCKWAVE DESIGN

Approach
- Prediction-based approach that captures long-term fairness and efficiency
- Mechanism for allocations that accounts for future rounds

Key Design Components
- Bayesian model to predict utility under dynamic adaptation
- Input to a market-based solver that gives us provable guarantees.
SHOCKWAVE: PREDICTING DYNAMIC ADAPTATION

Challenge: Predict utility of future epochs in presence of dynamic adaptation
Challenge: Predict utility of future epochs in presence of dynamic adaptation

**Handful of patterns** used for dynamic adaptation

- **Accordion**: Alternating batch sizes over epochs
- **GNS**: Doubling batch size over epochs
SHOCKWAVE: PREDICTING DYNAMIC ADAPTATION

Challenge: Predict utility of future epochs in presence of dynamic adaptation

Handful of patterns used for dynamic adaptation

Accordion: Alternating batch sizes over epochs
GNS: Doubling batch size over epochs

Approach: Model dynamic adaptation as a mixture of regimes
Use profiles for each batch size regime
SHOCKWAVE: PREDICTING DYNAMIC ADAPTATION

Mixture of regimes with different batch sizes

✓ Example - Job i 100 epochs:
  30 epochs (30%) for BS-512,
  70 epochs (70%) for BS-4096
Mixture of regimes with different batch sizes

- Example - Job $i$ 100 epochs:
  - 30 epochs (30%) for BS-512,
  - 70 epochs (70%) for BS-4096

Use a Dirichlet prior to model the mixture

- Job $i$ Prior: Dir(50%, 50%): 50 epochs BS512,
  - 50 epochs BS-4096

Observe batch scaling and update the posterior

- Job $i$ Posterior: Dirichlet (65%, 35%): 65 epochs for BS512, 35 epochs for BS-4096
Mixture of regimes with different batch sizes

- Example - Job $i$ 100 epochs:
  - 30 epochs (30%) for BS-512,
  - 70 epochs (70%) for BS-4096

Use a Dirichlet prior to model the mixture

- Job $i$ Prior: $\text{Dir}(50%, 50%)$: 50 epochs BS512,
  - 50 epochs BS-4096

Observe batch scaling and update the posterior

- Job $i$ Posterior: Dirichlet (65%, 35%): 65 epochs
  for BS512, 35 epochs for BS-4096

Temporal-dependence across epochs

Restatement rule: only updates parameters for completed epochs
SHOCKWAVE: DYNAMIC MARKETS

Goal: Scheduling policy that accounts for the past and future utilities

Market theory: Provable guarantees for efficiency, fairness.
SHOCKWAVE: DYNAMIC MARKETS

Goal: Scheduling policy that accounts for the past and future utilities

Market theory: Provable guarantees for efficiency, fairness.

Static market: Every training job has a known, time-invariant utility $U(x)$
  - Utility $U(x)$: map allocated GPU-time to training throughput for a job

Volatile Fisher Market (VFM) in Shockwave
  - Operate at discrete time intervals (rounds)
  - Every training job has a time-variant utility $U_t(x)$ for each round $(t)$
  - Solve for allocation that leads to market equilibrium
SHOCKWAVE: DYNAMIC MARKET PROPERTIES

Maximizes Nash Social Welfare over time

\[ U_i(X_i) = \sum_t u_{it}(x_{it}) \]

\[ \text{NSW}_{OT}(U_1(X_1), \ldots, U_N(X_N)) = \prod_i U_i(X_i)^{\frac{B_i}{\sum_i B_i}} \]

Maximizes cluster-wide utility

Efficiency
SHOCKWAVE: DYNAMIC MARKET PROPERTIES

Maximizes Nash Social Welfare over time

\[ U_i(X_i) = \sum_t u_{it}(x_{it}) \]

Minimizes product of FTF over time

Minimize \[ \prod_i \rho_i \]

\[ \rho_i = \frac{(\text{Finish time in a shared cluster})}{(\text{Finish in an } 1/N \text{ exclusive cluster})} \]

Maximizes cluster-wide utility

\[ \text{NSW}_{\text{OT}}(U_1(X_1), \ldots, U_N(X_N)) = \prod_i U_i(X_i)^{\frac{B_i}{\sum_i B_i}} \]

Provides sharing incentive

Efficiency

Fairness
SHOCKWAVE ARCHITECTURE

(1) Job Arrival
(2) Job Batch Size Change Trigger
(3) Posterior Update

Job J1:
- Training Progress
- Batch Size History

Jobs J2, J3:

Bayesian Predictive Model for Dynamic Batch Size Scaling
SHOCKWAVE ARCHITECTURE

1. Job Arrival
2. Job Batch Size Change Trigger
3. Posterior Update

- Bayesian Predictive Model for Dynamic Batch Size Scaling
  - Predicted Batch Size
  - Long-term Efficiency (Makespan) Estimator
  - Long-term Fairness Estimator
  - Makespan Estimate
  - FTF Estimate

4. Schedule at round t
5. Schedule at round t+1
6. Schedule at round t+T-1

Solver (Nash Social Welfare Optimizer)
Testbeds
- 32 Nvidia Quadro RTX5000 GPUs (16GB) at TACC
- Simulated clusters with 64-256 GPUs
- High simulation fidelity (<5% deviation)

Workloads
- Gavel Workload Generator (OSDI’20)
- Microsoft’s Philly trace (ATC’19)
- Mix of GNS and Accordion jobs (50%-50%)

EVALUATION

<table>
<thead>
<tr>
<th>Model</th>
<th>Task</th>
<th>Dataset</th>
<th>Batch Size(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>Image Classification</td>
<td>ImageNet</td>
<td>16 - 128</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>Image Classification</td>
<td>CIFAR-10</td>
<td>16 - 256</td>
</tr>
<tr>
<td>LSTM</td>
<td>Language Modeling</td>
<td>Wikitext-2</td>
<td>5 - 80</td>
</tr>
<tr>
<td>Transformer</td>
<td>Language Translation</td>
<td>Multi30k(DE-EN)</td>
<td>16 - 256</td>
</tr>
<tr>
<td>Recoder Autoencoder</td>
<td>Recommendation</td>
<td>ML-20M</td>
<td>512 - 8192</td>
</tr>
</tbody>
</table>
END-TO-END COMPARISONS

Gavel  Themis  AlloX  Shockwave

32-GPU Cluster in TACC, Gavel trace
End-to-end comparisons

Reduces Makespan by ~1.3x, Unfair fraction by ~2x

32-GPU Cluster in TACC, Gavel trace
32-GPU Cluster in TACC, Gavel trace

END-TO-END COMPARISONS

Reduces Makespan by ~1.3x, Unfair fraction by ~2x, maintains JCT
COMPARING SCHEDULERS: EFFICIENCY

(X)Small (35%)  Small (37%)  Medium (20%)  (X)Large (8%)

CPU #

Round
COMPARING SCHEDULERS: EFFICIENCY

- (X)Small (35%)
- Small (37%)
- Medium (20%)
- (X)Large (8%)

**Shockwave**

**Allox**

**Gavel**
COMPARING SCHEDULERS: EFFICIENCY

Shockwave minimizes Makespan while lowering the number of jobs with unfair allocations.
CONCLUSION

Dynamic Adaptation introduces challenges for fairness and efficiency

Shockwave: New predictive scheduler that uses a Market-based formulation

  Bayesian model for predicting dynamic adaptation
  Time-varying utility and provable guarantees on fairness and efficiency

Open source: https://github.com/uw-mad-dash/shockwave

Questions?