Parity Models
Erasure-Coded Resilience for Prediction Serving Systems

Jack Kosaian  Rashmi Vinayak  Shivaram Venkataraman
Machine learning lifecycle

**Training**
- Get a model to reach desired accuracy
- “Batch” jobs
- Hours to weeks

**Inference**
- Deploy model in target domain
- Online
- Milliseconds
Machine learning inference

queries → predictions

0.15 0.8 0.05

cat dog bird
Prediction serving systems

Inference in datacenter/cluster settings

Open Source

Cloud Services

Amazon SageMaker
Prediction serving system architectures

queries -> Frontend

Frontend -> predictions

model instances
Machine learning inference

- translation
- question-answering
- ranking

Must operate with low, predictable latency
Unavailability in serving systems

- Slowdowns and failures (unavailability)
  - Resource contention
  - Hardware failures
  - Runtime slowdowns
  - ML-specific events

- Result in **inflated tail latency**
  - Cause prediction serving systems to miss SLOs

Must alleviate slowdowns and failures
Redundancy-based resilience

- **Proactive**: send each query to 2+ servers
- **Reactive**: wait for a timeout before duplicating query
Erasure codes: proactive, resource-efficient

Relation to \((n, k)\) notation

\[ n = k + r \]

\[ \begin{align*}
D_1 & \quad \text{encoding} \quad \text{r “parity” units} \\
D_2 & \quad \text{“parity” } P = D_1 + D_2
\end{align*} \]

\[ D_2 = P - D_1 \]

any \(k\) out of \((k+r)\) units → decoding → original \(k\) data units
Erasure codes: proactive, resource-efficient

- Recovery Delay (lower is better)
- Resource Overhead (lower is better)

- Reactive
- Proactive

- Storage
- Communication
- Prediction Serving Systems

Erasure codes: proactive, resource-efficient
Coded-computation

Our goal: Using erasure codes to reduce tail latency in prediction serving

Goal: preserve results of computation over queries

\[ F(X_1) \quad F(X_2) \]

queries

\[ F \quad F \quad F \]

models

\[ F(X_1) \quad F(X_2) \]

predictions
Coded-computation

Our goal: Using erasure codes to reduce tail latency in prediction serving

Encode queries

\[ F(X_1) \quad F(X_2) \]
Coded-computation

Our goal: Using erasure codes to reduce tail latency in prediction serving

Decode results of inference over queries

\[ F(X_1) \quad \text{encode} \quad F \quad \text{decode} \quad F(X_2) \]

\[ F(X_2) \quad \text{encode} \quad F \quad \text{decode} \quad F(P) \]

“parity query”
Traditional coding vs. coded-computation

Codes for storage

\[ D_1 \quad D_2 \quad \text{encode} \quad \text{decode} \quad D_2 \]

Coded-computation

\[ X_1 \quad X_2 \quad \text{encode} \quad \text{decode} \quad F(X_2) \]

Need to recover computation over inputs
Challenge: Non-linear computation

Linear computation
Example: \( F(X) = 2X \)

\[
\begin{align*}
X_1 & \rightarrow 2X \\
X_2 & \rightarrow 2X \\
P &= X_1 + X_2 \\
F(X_2) &= F(P) - F(X_1) \\
&= 2(X_1 + X_2) - X_1 \\
&= 2X_2
\end{align*}
\]

Non-linear computation
Example: \( F(X) = X^2 \)

\[
\begin{align*}
X_1 & \rightarrow X^2 \\
X_2 & \rightarrow X^2 \\
P &= X_1 + X_2 \\
F(X_2) &= F(P) - F(X_1) \\
&= 2(X_1 + X_2)^2 - X_1^2 \\
&= X_2^2 + 2X_1X_2 \\
&\times \\
\text{Actual is } X_2^2
\end{align*}
\]
Challenge: Non-linear computation

Linear computation
Example: \( F(X) = 2X \)

\[
X_1 \quad X_2 \quad P = X_1 + X_2
\]

\[
F(X_2) = F(P) - F(X_1) = 2(X_1 + X_2) - X_1 = 2X_2
\]

Non-linear computation

\[
X_1 \quad X_2 \quad P = X_1 + X_2
\]

\[
F(X_2) = F(P) - F(X_1) = ???
\]
Current approaches to coded-computation

• Lots of great work on **linear computations**
  • Huang 1984, Lee 2015, Dutta 2016, Dutta 2017, Mallick 2018, more…

• Recent work supports restricted **nonlinear computations**
  • Yu 2018
  • At least 2x resource overhead

**Current approaches insufficient for neural networks in prediction serving systems**
Our approach:
Learning-based coded-computation

Learning a Code: Machine Learning for Approximate Non-Linear Coded Computation
https://arxiv.org/abs/1806.01259

Parity Models: Erasure-Coded Resilience for Prediction Serving Systems
To appear in ACM SOSP 2019
https://jackkosaian.github.io
Learning an erasure code?

Design encoder and decoder as neural networks

Learning a Code: Machine Learning for Approximate Non-Linear Coded Computation
https://arxiv.org/abs/1806.01259
Learning an erasure code?

Design encoder and decoder as neural networks

![Diagram showing encoder and decoder with annotations]

Learning a Code: Machine Learning for Approximate Non-Linear Coded Computation

https://arxiv.org/abs/1806.01259
Learn computation over parities

Use simple, fast encoders and decoders
Learn computation over parities: “parity model”

Parity Models: Erasure-Coded Resilience for Prediction Serving Systems
To appear in ACM SOSP 2019
https://jackkosaian.github.io
Designing parity models

Parity model goal

Transform parities such that decoder can reconstruct unavailable predictions

\[ P = X_1 + X_2 \]

\[ F(X_2) = F_P(P) - F(X_1) \]
Designing parity models

Parity model goal
Transform parities such that decoder can reconstruct unavailable predictions

\[ P = X_1 + X_2 \]

\[ F(X_2) = F_P(P) - F(X_1) \]

\[ F_P(P) = F(X_1) + F(X_2) \]

Learn a parity model
Designing parity models

Parity model goal
Transform parities such that decoder can reconstruct unavailable predictions

\[
X_1 \quad X_2 \quad P = X_1 + X_2
\]

parity model \( (F_P) \)

\[
F(X_2) = F_P(P) - F(X_1)
\]

\[
F_P(P) = F(X_1) + F(X_2)
\]
Training a parity model

1. Sample k inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropagate loss
5. Repeat

\[ P = X_1 + X_2 \]

Desired output: \( F(X_1) + F(X_2) \)

\[
\begin{align*}
0.8 & \quad 0.15 & \quad 0.05 \\
0.2 & \quad 0.7 & \quad 0.1
\end{align*}
\]
Training a parity model

1. Sample $k$ inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropogate loss
5. Repeat
1. Sample k inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropogate loss
5. Repeat

Desired output: $F(X_1) + F(X_2)$

$P = X_1 + X_2$

$F_P(P)_3$

$0.03 0.02 0.95$

$0.3 0.3 0.4$
Training a parity model: higher parameter $k$

1. Sample inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropagate loss
5. Repeat

Desired output: $F(P) = F(X_1) + F(X_2) + F(X_3) + F(X_4)$
Training a parity model: different encoder

1. Sample inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropogate loss
5. Repeat

\[ F_{P}(P)_{3} \]
Learning results in approximate reconstructions

Appropriate for machine learning inference

1. Predictions resulting from inference are approximations
2. Inaccuracy only at play when predictions otherwise slow/failed
Implementing parity models in Clipper
Design space in parity models framework

Encoder/decoder
- Many possibilities
- Generic: addition/subtraction
- Can specialize to task

Parity model architecture
- Again, many possibilities
- Same as original model ⇒ same latency as original

\[
P = X_1 + X_2
\]

\[
F(X_2) = F_P(P) - F(X_1)
\]
Evaluation

1. How accurate are reconstructions using parity models?

2. How much can parity models help reduce tail latency?
Evaluation of Accuracy

- Addition/subtraction code
- $k = 2$, $r = 1$ ($P = X_1 + X_2$)
- 2x less overhead than replication
Parity model only comes into play when predictions are slow/failed

- Addition/subtraction code
- $k = 2, r = 1 (P = X_1 + X_2)$
- 2x less overhead than replication
Evaluation of Accuracy

Parity model only comes into play when predictions are slow/failed

- Addition/subtraction code
- $k = 2, r = 1 \ (P = X_1 + X_2)$
- 2x less overhead than replication
Evaluation of Overall Accuracy

Parity model only comes into play when predictions are slow/failed

- Addition/subtraction code
- $k = 2$, $r = 1$ ($P = X_1 + X_2$)
- 2x less overhead than replication
Evaluation of Overall Accuracy

Parity model only comes into play when predictions are slow/failed

- Addition/subtraction code
- $k = 2$, $r = 1$ ($P = X_1 + X_2$)
- 2x less overhead than replication

![Graph showing overall accuracy comparison between original model and parity model.](image-url)
Evaluation of Overall Accuracy

Parity model only comes into play when predictions are slow/failed

- Addition/subtraction code
- $k = 2, \ r = 1 \ (P = X_1 + X_2)$
- 2x less overhead than replication

Expected operating regime
Evaluation of Accuracy: Higher values of $k$

Tradeoff between resource-overhead, resilience, and accuracy

- Addition/subtraction code
Evaluation of Accuracy: Object-localization

- Ground Truth
- Available
- Parity Models
Evaluation of Accuracy: Task-specific encoder

22% accuracy improvement over addition/subtraction at $k = 4$
Evaluation of Tail Latency Reduction: Setup

• Implemented in Clipper prediction serving system

• Evaluate with 18-36 nodes on AWS with varying:
  • Inference hardware (GPUs, CPUs)
  • Query arrival rates
  • Batch sizes
  • Levels of load imbalance
  • Amounts of redundancy
  • Baseline approaches

• Baseline: approach with same number of resources as parity models
Evaluation of Tail Latency Reduction

In presence of resource contention

- **Parity Models**
- **Equal-Resources**

Latency (ms):
- **Median**: 20
- **Mean**: 18
- **99th**: 30
- **99.5th**: 32
- **99.9th**: 40

*same median*

40% reduction
Limitations of current parity models framework

• Training a parity model is slow!
  • Dataset with N samples ⇒ parity model dataset with $N^k$ samples
Training a parity model

1. Sample k inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropogate loss
5. Repeat

\[ F(X_1) + F(X_2) \]

\[ P = X_1 + X_2 \]

\[ F_P(P)_3 \]

\{ compute loss \}
Limitations of current parity models framework

• Training a parity model is slow!
  • Dataset with N samples ⇒ parity model dataset with $N^k$ samples
  • How to efficiently train under this combinatorial explosion?

• Theoretical understanding?
  • Subject to same problems as existing NNs (e.g., adversarial examples)
  • Can’t bound inaccuracy

• Potential privacy concerns
  • Combining query A with query B into a parity query might leak info

• More research needed to tackle the above
Landscape of learning in coded-computation

Learn a code

\[ X_1 \xrightarrow{\text{encoder}} \text{decoder} \]

Learning a parity model

\[ P = X_1 + X_2 \]

\[ F(X_2) = F_P(P) - F(X_1) \]
Jointly learn encoders, decoders, and parity models?

Balance complexity, execution time across components

Landscape of learning in coded-computation
Parity Models: Erasure-Coded Resilience for Prediction Serving Systems

• Coded-computation is promising, but current approaches cannot support popular machine learning models like neural networks

• Parity models: judicious use of learning allows for accurate reconstruction of unavailable ML inference predictions

• Enables erasure-coded resilience in prediction serving systems

Code available: github.com/Thesys-lab/parity-models