Computational Polarization and Straggler Resilient Serverless Optimization

Mert Pilanci
March 27, 2020

Electrical Engineering
Stanford University
Annual Conference on Information Sciences and Systems (CISS) 2020
Every day, we create 2.5 billion gigabytes of data
Data stored grows $4x$ faster than world economy (Mayer-Schonberger)
Deep learning revolution

ImageNet Classification, top-5 error (%)

<table>
<thead>
<tr>
<th>Year</th>
<th>Model</th>
<th>Layers</th>
<th>Top-5 Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILSVRC 2010</td>
<td>NEC America</td>
<td></td>
<td>28.2</td>
</tr>
<tr>
<td>ILSVRC 2011</td>
<td>Xerox</td>
<td></td>
<td>25.8</td>
</tr>
<tr>
<td>ILSVRC 2012</td>
<td>AlexNet</td>
<td>8</td>
<td>16.4</td>
</tr>
<tr>
<td>ILSVRC 2013</td>
<td>Clarif</td>
<td>8</td>
<td>11.7</td>
</tr>
<tr>
<td>ILSVRC 2014</td>
<td>VGG</td>
<td>19</td>
<td>7.3</td>
</tr>
<tr>
<td>ILSVRC 2014</td>
<td>GoogleNet</td>
<td>22</td>
<td>6.7</td>
</tr>
<tr>
<td>ILSVRC 2015</td>
<td>ResNet</td>
<td>152</td>
<td>3.5</td>
</tr>
</tbody>
</table>
in machine learning and data science

○ more data results in better and accurate models
  → large scale distributed computing problems
in machine learning and data science

- more data results in better and accurate models
  → **large scale distributed computing problems**

Can we scale computation inexpensively?

Our approach: **serverless systems with error correction to auto-scale computation**


B. Bartan and M. Pilanci *Straggler Resilient Serverless Computing Based on Polar Codes*, Allerton 2019
Distributed computation

DATA

4
Error Resilient Matrix Multiplication

Speeding Up Distributed Machine Learning Using Codes. Lee et al., 2017
Error Resilient Matrix Multiplication

Speeding Up Distributed Machine Learning Using Codes. Lee et al., 2017
Serverless computing: AWS Lambda

- Low cost, no upfront investment
- 900 seconds single-core, 3GB RAM
- Python, Java, C#
Serverless computing: AWS Lambda

- Lambda functions are *stateless*
  
  Local file system access and child processes may not extend beyond the lifetime of the request
  
  Persistent state should be stored in a storage service (e.g., S3)

- Pywren (E. Jonas et al., 2017)

- Google Cloud and Microsoft Azure offer similar services
Serverless computing: AWS Lambda

○ return times

![Diagram showing compute latency and number of workers]
Polar Codes were invented by Arikan in 2009

- Combines communication channels recursively to obtain better/worse channels
- It is the first code with an explicit construction to provably achieve the channel capacity for all symmetric discrete memoryless channels
- 3rd Generation Partnership Project (3GPP) adopted polar codes as the official coding scheme for the control channels of the 5G New Radio interface.
Polar Codes: Recursive Channel Transformation

2×2 construction

4×4 construction
Computational Polar Codes

\[ A \xrightarrow{} \text{worker} \xrightarrow{} f(A) \]
Computational Polar Codes

\[ A_1 \xrightarrow{} \text{Worker 1} \xrightarrow{} f(A_1) \]

\[ A_2 \xrightarrow{} \text{Worker 2} \xrightarrow{} f(A_2) \]
Hadamard transform

\[ A_1 \rightarrow + \rightarrow A_1 + A_2 \]

\[ A_2 \rightarrow \times \rightarrow -1 \rightarrow + \rightarrow A_1 - A_2 \]
Butterfly coded computation

\[
\begin{align*}
A_1 & \xrightarrow{\times} A_2 \\
\quad & \xrightarrow{+} A_1 + A_2 \\
& \xrightarrow{\text{worker 1}} f(A_1 + A_2)
\end{align*}
\]

\[
\begin{align*}
A_2 & \xrightarrow{-1} A_1 - A_2 \\
\quad & \xrightarrow{+} A_1 - A_2 \\
& \xrightarrow{\text{worker 2}} f(A_1 - A_2)
\end{align*}
\]
Decoding the original computation \( f(A_1) \) and \( f(A_2) \) for linear functions
Runtime distribution

\[ A_1 \times A_2 \rightarrow T_2 \rightarrow A_1 - A_2 \rightarrow f(A_1 - A_2) \rightarrow \text{worker 2} \]

\[ T_1 \rightarrow A_1 + A_2 \rightarrow f(A_1 + A_2) \rightarrow \text{worker 1} \]

\[ \text{decode } f(A_1) \]

\[ \text{max}(T_1, T_2) \rightarrow f(A_1) \]

\[ \text{decode } f(A_2) \]

\[ \text{min}(T_1, T_2) \rightarrow f(A_2) \]

\[ T_1 \rightarrow \max(T_1, T_2) \rightarrow T_2 \rightarrow \min(T_1, T_2) \]
4 by 4 construction
Computational Polarization Process

\[ \max(T_1, T_2) \quad \min(T_1, T_2) \quad \max(T_3, T_4) \quad \min(T_3, T_4) \]

\[ \max(\max(T_1, T_2), \max(T_3, T_4)) \quad \min(\max(T_1, T_2), \max(T_3, T_4)) \quad \max(\min(T_1, T_2), \min(T_3, T_4)) \quad \min(\min(T_1, T_2), \min(T_3, T_4)) \]
Computational Polarization Process

- Functional Martingale process
- $F(t)$ is the cumulative density function of the i.i.d. run-times
  
  $F_{n+1}(t) = \begin{cases} 
  1 - (1 - F_n(t))^2 & \text{with probability } \frac{1}{2} \\
  F_n(t)^2 & \text{with probability } \frac{1}{2} 
  \end{cases}$

- $\mathbb{E}[F_{n+1}(t) | F_n] = F_n(t)$
- **Theorem:** $\|F_{n+1}(t) - F_n(t)\|_{L_2} \to 0$ as $n \to \infty$ with rate $O(2^{-2\sqrt{n}})$

$F_n(t)$ converges to unit step functions
run-time distributions converge to the Dirac measure

Run-time distributions
Fixing certain inputs to zero

\[ A_1, A_2, A_3, A_4 \]

\[ M_1: f(A_1 + A_2) + f(A_3 + A_4) \]

\[ M_2: f(A_1 + A_2) - f(A_3 + A_4) \]

\[ M_3: f(A_1 - A_2) + f(A_3 - A_4) \]

\[ M_4: f(A_1 - A_2) - f(A_3 - A_4) \]
Fixing certain inputs to zero
Computational Polarization

(B. Bartan and M. Pilanci, Straggler Resilient Serverless Computing Based on Polar Codes, 2019)
Comparison with other coding methods

- Reed-Solomon codes, LT codes, LDPC codes, Fermat Number Transform (FNT) based codes
- Computational Polar codes have $O(n \log n)$ encoding and decoding complexity
- only addition and subtraction operations in encoding and decoding
- can scale to 10,000 workers
Compute jobs
Elastic computing

- AWS Lambda serverless compute jobs 1.5 GB memory each
  
  (a) uncoded: 500 workers
  
  (b) coded: 1500 workers (1000 redundant parity)
Computational Polarization for optimization on AWS Lambda

- encode data matrix $A$ for gradient calculation, e.g., $Ax$ and $A^T y$ for Least Squares and Generalized Linear Models

random data ($20000 \times 4800$) Imagenet ($2013526 \times 196608 \sim 1.2$ TB)

![Graph showing the comparison between no coding (128 workers) and coding (256 workers) for different datasets and computational times]
Computing Nonlinear Functions

- linear functions of data $f(A)$
- polynomial functions of data $f(A)$
- gradient and Hessian calculations involving data $A$
Computational Polarization for gradient estimation

○ gradient estimator

\[ \frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + h e_i) - f(x - h e_i)}{2h} \]

○ coded gradient estimator

\[ \frac{f(x + h z_i) - f(x - h z_i)}{2h} \]

○ \( z_i \): redundant function evaluation directions =

○ decode the gradient \( \frac{\partial f(x)}{\partial x} \) from \( \langle \frac{\partial f(x)}{\partial x}, z_i \rangle \)

B. Bartan, M. Pilanci, Distributed Black-Box Optimization via Error Correcting Codes, 2019
Adversarial Examples

- given a trained neural network
- constrained optimization problem

$$\min_x ||x - x_0|| \text{ subject to } \text{probability}_j(x) > \text{probability}_i(x)$$

(Szegedy et al., 2014, Goodfellow et al., 2015)
Comparison with finite differences and random search

- plane classified as truck in CIFAR10

<table>
<thead>
<tr>
<th></th>
<th>FD</th>
<th>H w/ decoding</th>
<th>ES w/ H</th>
<th>HD w/ decoding</th>
<th>ES w/ HD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>time (ms)</strong></td>
<td>0</td>
<td>20000</td>
<td>40000</td>
<td>60000</td>
<td>80000</td>
</tr>
<tr>
<td><strong>cost</strong></td>
<td>1000</td>
<td>0.1</td>
<td>ex_no</td>
<td>10</td>
<td>t = 1</td>
</tr>
</tbody>
</table>

1000 iters with c = 0.1, ex_no = 10, t = 1

Choromanski et al. Structured evolution with compact architectures for scalable policy optimization, 2018
Conclusions and future work

✓ Scalable and error resilient distributed computing system
✓ cheap encoding and decoding
✓ distributed Least Squares and GLMS
→ privacy and encryption
→ more general convex optimization problems with constraints, e.g.,
convex optimization for neural networks

M. Pilanci, T. Ergen

stanford.edu/~pilanci


- M. Pilanci, T. Ergen, Neural Networks are Convex Regularizers: Exact Polynomial-time Convex Optimization Formulations for Two-Layer Networks, ICML 2020