# Near-Data Processing for Differentiable Machine Learning Models

Hyeokjun Choe<sup>1</sup>, Seil Lee<sup>1</sup>, Hyunha Nam<sup>1</sup>, Seongsik Park<sup>1</sup>, Seijoon Kim<sup>1</sup>, Eui-Young Chung<sup>2</sup> and Sungroh Yoon<sup>1,3\*</sup>

<sup>1</sup>Electrical and Computer Engineering, Seoul National University

<sup>2</sup>Electrical and Electronic Engineering, Yonsei University

<sup>3</sup>Neurology and Neurological Sciences, Stanford University

\*Correspondence: sryoon@snu.ac.kr

Homepage: http://dsl.snu.ac.kr

May 19th, 2017

# Outline

#### 1 Introduction

- 2 Background
- 3 Proposed Methodology
- 4 Experimental Results
- **5** Discussion and Conclusion

# Outline

#### 1 Introduction

- 2 Background
- 3 Proposed Methodology
- 4 Experimental Results
- 5 Discussion and Conclusion

# Machine Learning's Success

- Big data
- Powerful parallel processors
- $\Rightarrow$  Sophisticated models



Source: http://ml.cecs.ucl.edu/

### Issues on Conventional Memory Hierachy

- Data movement in memory hierarchy
  - Computational efficiency  $\Downarrow$
  - Power consumption  $\uparrow$



http://computerscience.chemeketa.edu/cs160Reader/ComputerArchitecture/MemoryHeirarchy.html

# Near-data Processing (NDP)

- Memory or storage with intelligence (i.e., computing power)
- Process the data stored in memory or storage
- Reduce the data movements, CPU offloading



# $\operatorname{ISP-ML}$

- ISP-ML: a full-fledged ISP-supporting SSD platform
- Easy to implement machine learning algorithm in C/C++
- For validation, three SGD algorithms were implemented and experimented with ISP-ML



# Outline

#### 1 Introduction

# 2 Background

3 Proposed Methodology

#### 4 Experimental Results

#### 5 Discussion and Conclusion

# Machine Learning as an Optimization Problem

- Machine learning categories
  - Supervised learning, unsupervised learning, reinforcement learning
- The main purpose of supervised machine learning
  - Find the optimal  $\boldsymbol{\theta}$  that minimizes  $F(D; \boldsymbol{\theta})$

$$F(D, \boldsymbol{\theta}) = L(D, \boldsymbol{\theta}) + r(\boldsymbol{\theta})$$
(1)



- D : input data
- $\theta$  : model parameters
- L : loss function
- $r \ : \ {\rm regularization \ term}$
- F : objective function

# Gradient Descent



- learning rate
- t : iteration index
- i: data sample index
- 1st-order iterative optimization algorithm
  - Use all samples per iteration
- Stochastic gradient descent (SGD)
  - Use only one sample per iteration.
- Minibatch stochastic gradient descent
  - Between gradient descent and SGD
  - Use multiple samples per iteration

# Parallel and Distributed SGD

- Synchornous SGD
  - Parameter server aggregates  $\nabla \boldsymbol{\theta}_{slave}$  synchronously.
- Downpour SGD
  - Workers communicate with parameter server asynchronously.
- Elastic Average SGD (EASGD)
  - Each worker has own parameters
  - Workers transfer  $(\boldsymbol{\theta}_{slave} \boldsymbol{\theta}_{master})$ , not  $\nabla \boldsymbol{\theta}_{slave}$



|--|

# Fundamentals of Solid-State Drives (SSDs)

- SSD Controller
  - Embedded processor for FTL
    - HDD emulation
    - Wear Leveling, Garbage collection, etc.
  - Cache controller
  - Channel controller
- DRAM
  - Cache and Buffer
  - 512MB 2GB
- NAND flash arrays
  - Simultaneously accessible
- Host interface logic
  - SATA, PCIe





#### Previous Work on Near-Data Processing:PIM

- Perform computation inside the main memory
- 3D stacked memory (e.g. HMC) is used for PIM recently
  - Implement processing unit in Logic Layer
- Applications: sorting, string matching, CNN, matrix multiplication etc.



Source: Pawlowski, J. Thomas. "Hybrid memory cube (HMC)." Hot Chips 23 Symposium (HCS), 2011 IEEE. IEEE, 2011.

# Previous Work on Near-Data Processing:ISP

- Perform computation inside the storage
- ISP with embedded processor
  - Pros: easy to implement, flexible
  - Cons: no parallelism
- ISP with dedicated hardware logic
  - Pros: channel parallelism, hardware acceleration
  - Cons: hard to implement and change
- Applications: DB query (scan, join), linear regression, k-means, string match etc.

# Outline

#### 1 Introduction

- 2 Background
- 3 Proposed Methodology
  - 4 Experimental Results
- 5 Discussion and Conclusion

# ISP-ML: ISP Platform for Machine Learning on SSDs

- ISP-supporting SSD simulator
  - Implemented in SystemC on the Synopsys Platform Architect
    - Software/Hardware co-simulation
    - Easily executes various machine learning algorithms in C/C++
- Transaction level simulator
  - For reasonable simulation speed
- ISP components
  - ISP SW, ISP HW





# ISP-ML: ISP Platform for Machine Learning on SSDs

- We implemented two types of ISP hardware components.
  - Channel controller: perform primitive operations on the stored data.
  - Cache controller: collect the results from each of the channel controller.
- Master-slave architecture
- They communicate with each other.



Algorithm 1 ISP-Based Synchro, SGD Algorithm 3 ISP-Based EASGD Algorithm 2 ISP-Based Downpour SGD 1: Read page-sized data D1 from NAND array 1: Read page-sized data D1 from NAND array 1: Read page-sized data D, from NAND array b i: channel controller index > i: channel controller index p il channel controller index > j: NAND flash page index > j: NAND flash page index > j: NAND flash page index > k: training sample index (within a minibatch) > k: training sample index (within a minibatch) > k; training sample index (within a minibatch) 2: Pull  $\theta_{carbo}$  from the cache controller buffer 2: Pull  $\theta_{eacher}$  from the cache controller buffer 2:  $k \leftarrow 0$ 3:  $\theta^i \leftarrow \theta_{cuche}$ 3: 0° ← 0 eacho 3: while k < b do p b; minibatch size 4:  $\Delta \theta^i \leftarrow 0, k \leftarrow 0$  $4: \Delta \theta^{i} \leftarrow 0, k \leftarrow 0$ 4: Calculate  $F(D^i_{ik}, \theta^i)$ temp  $\leftarrow$  temp  $+ \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 5: while k < b do a b: minibatch size 5: while k < b do b b: minibatch size 6:  $k \leftarrow k+1$ Calculate  $F(D_{ik}^i, \theta^i)$ Calculate F(D', , 0') 6: 6: 7: end while  $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{+*}, \theta^i)$  $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D^{i}_{ik}, \theta^{i})$ 7: 7: 8:  $\theta^{i} \leftarrow \theta^{i} - l \text{ temp}$ 8.  $k \leftarrow k+1$ 8:  $k \leftarrow k + 1$ 9: if  $j \mod \tau = 0$  then > mod: modulus 9: end while 10: Pull  $\theta_{cache}$  from the cache controller buffer 9: end while 11: 10: Push  $\Delta \theta^s$  and wait temp  $\leftarrow \theta_{cache}$ 10: if  $i \mod \tau = 0$  then > mod: modulus  $\Delta \theta^{i} \leftarrow \alpha(\theta^{i} - \text{temp})$ 12. > Lines 11-12: executed by the cache controller HE: Push  $\Delta \theta^{i}$ 13:  $\theta^{i} \leftarrow \theta^{i} - \Delta \theta^{i}$ 11:  $\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{2} \sum_{i} \Delta \theta^{i}$  $\theta_{cache } \leftarrow \theta_{cache } - \Delta \theta^{*}$ to by cache ctrl. 14: Push  $\Delta \theta^{\dagger}$ 12: Signal each channel controller 13: end if  $\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} + \Delta \theta^{1}$ > by cache ctrl. 16: end if



Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
1: Read page-sized data $D_{i}^{b}$ from NAND array b; t. channel controlser more b; t. NAND link page index b; t. training sample index (within a minibatch) 2: Pull $\theta_{eache}$ from the cache controller buffer 3: $\theta^{i} \leftarrow \theta_{eache}$ 4: $\Delta \theta^{i} \leftarrow 0$ , $k \leftarrow 0$ 5: while $k < b$ do 6: Calculate $F(D_{jk}^{i}, \theta^{i})$ 7: $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D_{jk}^{i}, \theta^{i})$	1: Read page-sized data $D_2^{ij}$ from NAND array $\triangleright$ : channel controller index $\triangleright$ : $j$ : NAND flash page index $\triangleright$ : $k$ : training sample index (within a minibatch) 2: Pull $\theta_{eacher}$ from the cache controller baffer $3: \theta^i + \theta_{eacher}$ $4: \Delta \theta^i + 0, \ k \leftarrow 0$ 5: while $k < b$ do $\triangleright$ is minibatch size 6: Calculate $F(D_{2k}^{ij}, \theta^i)$ 7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D_{2k}^{ij}, \theta^i)$	1: Read page-sized data $D_{j}^{i}$ from NAND array $\triangleright$ i; channel controller index $\triangleright$ j; NAND flash page index $\triangleright$ k; training sample index (within a minibatch) 2: $k \leftarrow 0$ 3: while $k < b  do$ $\triangleright$ lo minibatch size 4: Calculate $F(D_{jk}^{i}, \theta^{i})$ 5: torr $\leftarrow + 1$ 7: end while 8: $\theta^{i} = \theta^{i} = 1$ form
8: $k \leftarrow k + 1$ 9: end while 10: Push $\Delta \theta^{*}$ and wait $\triangleright$ Lines 11-12: executed by the cache controller 11: $\left[ \theta_{cache} \leftarrow \theta_{cache} - \frac{1}{n} \sum_{i} \Delta \theta^{i} \right]$ 12: [Signal cach channel controller]	$\begin{array}{llllllllllllllllllllllllllllllllllll$	$ \begin{array}{llllllllllllllllllllllllllllllllllll$



Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
<ol> <li>Read page-sized data D<sup>1</sup><sub>2</sub> from NAND array b i: channel controller index b j: NAND flash page index b k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>1</sup><sub>2</sub> from NAND array</li> <li>▷ i: channel controller index</li> <li>▷ j: NAND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>f</sup><sub>4</sub> from NAND array</li> <li>▷ i; channel controller index</li> <li>▷ j; NAND flash page index</li> <li>▷ k; training sample index (within a minibatch)</li> </ol>
2: Pull $\theta_{cache}$ from the cache controller buffer	2: Pull $\theta_{\text{rache}}$ from the cache controller buffer	2: $k \leftarrow 0$
3: $\theta^i \leftarrow \theta_{cache}$	3: $\theta^* \leftarrow \theta_{eacho}$	3: while $k < b$ do $\Rightarrow b$ : minibatch size
4: $\Delta \theta^{i} \leftarrow 0,  k \leftarrow 0$	4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: Calculate $F(D_{jk}^i, \theta^i)$
5: while $k < b$ do $b$ : minibatch size	5: while $k < b$ do $b$ : minibatch size	5: temp $\leftarrow$ temp $+ \eta \nabla F(D_{jk}^{i}, \theta^{i})$
6: Calculate $F(D_{ik}^i, \theta^i)$	6: Calculate $F(D_{i,k}^{i}, \theta^{i})$	6: $k \leftarrow k+1$
7: $\Delta \theta^i \leftarrow \Delta \theta^i + n \nabla F(D^1, \theta^i)$	7: $\Delta \theta^i \leftarrow \Delta \theta^i \pm n \nabla F(D^i, \theta^i)$	7: end while
8: $k \leftarrow k+1$	8: $k \leftarrow k+1$	9: if $\tau \mod \tau = 0$ then p mode modulos
9: end while	9 end while	10: Pull $\theta_{rache}$ from the cache controller buffer
10: Push $\Delta \theta^{i}$ and wait	10: If $i \mod \pi = 0$ then to modules	11: temp $\leftarrow \theta_{cache}$
b. Lines 11-12: executed by the cache controller	II- Dark Ad <sup>1</sup>	12: $\Delta \theta^i \leftarrow \alpha(\theta^i - \text{temp})$
11. 0 0		13: $\theta^i \leftarrow \theta^i - \Delta \theta^i$
12: Signal auch changed controlled	12: $\theta_{cache} \leftarrow \theta_{cache} - \Delta \theta'$ is by cache ctri.	14: Push $\Delta \theta^{\prime}$
12. Signal each channel controller	13: end if	15: $[\theta_{cache} \leftarrow \theta_{cache} + \Delta \theta^*]$ > by cache ctrl.







Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
<ol> <li>Read page-sized data D<sup>1</sup><sub>2</sub> from NAND array b i: channel controller index</li> <li>j: NAND flash page index</li> <li>b: training sample index (within a minibatch)</li> <li>Pull θ<sub>cathle</sub> from the cache controller buffer</li> </ol>	<ol> <li>Read page-sized data D<sup>1</sup><sub>j</sub> from NAND array.</li> <li>i: channel controller index</li> <li>j: NAND flash page index</li> <li>j: training sample index (within a minibatch)</li> <li>Pull θ<sub>cacher</sub> from the cache controller baffer</li> </ol>	<ol> <li>Read page-sized data D<sup>1</sup><sub>2</sub> from NAND array         <ul> <li>i channel controller index</li> <li>j: NAND flash page index</li> <li>k: training sample index (within a minibatch)</li> </ul> </li> <li>k ← 0</li> </ol>
3: $\theta^i \leftarrow \theta_{cache}$	3: 0' + 0 eacho.	3: while $k < b$ do $> b$ : minibatch size
4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$ 5: while $k < b$ do $b$ : minibatch size 6: Calculate $F(D_{jk}^i, \theta^i)$ 7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D_{jk}^i, \theta^i)$	4: $\Delta \theta^i \leftarrow 0$ , $k \leftarrow 0$ 5: while $k < b \ do$ $b \ b$ : minibatch size 6: Calculate $F(D^i_{jk}, \theta^i)$ 7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{jk}, \theta^i)$	4: Calculate $F(D_{jk}^{i}, \theta^{i})$ 5: temp $\leftarrow$ temp $+ \eta \nabla F(D_{jk}^{i}, \theta^{i})$ 6: $k \leftarrow k + 1$ 7: end while 8: $\theta^{i} \leftarrow \theta^{i} - \frac{1}{h}$ temp
a: $\kappa \leftarrow \kappa + 1$ 9: end while	8: $k \leftarrow k + 1$ 9- and while	9: if $j \mod \tau = 0$ then $\triangleright \mod t$ mod: modulus 10: Pull $\theta_{cache}$ from the cache controller buffer
10: Push $\Delta \theta^{i}$ and wait	10: If $j \mod \tau = 0$ then $> \mod$ : modulus	11: temp $\leftarrow \theta_{cache}$
▷ Lines 11-12: executed by the cache controller 11: $[\theta_{\text{ouch}n} \leftarrow \theta_{\text{ouch}n} - \frac{1}{n} \sum_{i} \Delta \theta^{i}]$ 12: [Signal each channel controller]	$ \begin{array}{llllllllllllllllllllllllllllllllllll$	12: $\Delta \theta^{i} \leftarrow \alpha(\theta^{i} - \operatorname{temp})$ 13: $\theta^{i} \leftarrow \theta^{i} - \Delta \theta^{i}$ 14: Pash $\Delta \theta^{i}$ 15: $\theta_{\operatorname{cache}} \leftarrow \theta_{\operatorname{cache}} + \Delta \theta^{i}$ $\triangleright$ by cache ctrl. 16: end if



Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
1: Read page-sized data $D_{j}^{1}$ from NAND array $\flat :$ (channel controller index $\flat :$ (NAND link hyge index $\flat :$ (krithing sample index (within a minibatch) 2: Pall $\theta_{cache}$ from the cache controller buffer 3: $\theta^{i} + \theta_{eache}$ 4: $\Delta \theta^{i} \leftarrow 0,  k \leftarrow 0$ 5: while $k < b$ do 6: Calcular $F(D_{jk}^{i}, \theta^{i})$ 7: $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D_{jk}^{i}, \theta^{i})$	1: Read page-sized data $D_{j}^{1}$ from NAND array $\triangleright$ : channel controller index $\triangleright$ : $\beta$ : NAND flash page index (within a minibatch) 2: Pull $\theta_{cache}$ from the cache controller buffer 3: $\theta^{i} \leftarrow \theta_{-acheb}$ 4: $\Delta \theta^{i} \leftarrow 0$ , $k \leftarrow 0$ 5: while $k < b \ do$ $b \ b$ minibatch size 6: Calculate $F(D_{jk}^{i}, \theta^{i})$ 7: $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \nabla F(D_{jk}^{i}, \theta^{i})$	1: Read page-sized data $D_j^i$ from NAND array $\triangleright i$ ; channel controller index $\triangleright j$ ; NAND fash page index $\triangleright k$ ; training sample index (within a minibatch) 2: $k \leftarrow 0$ 3: while $k < b$ do $\triangleright b$ minibatch size 4: Calculate $F(D_{jk}^i, \theta^i)$ 5: temp $\leftarrow$ temp $+\eta \nabla F(D_{jk}^i, \theta^i)$ 6: $k \leftarrow k + 1$ 7: end while 8: $\theta^i \leftarrow \theta^i = 1$ temp
8: $k \leftarrow k + 1$ 9: and while	8: $k \leftarrow k+1$	9: if $j \mod \tau = 0$ then $\triangleright \mod$ : modelines 10: Pull $\theta$ from the cache controller buffer
$\begin{array}{l} \begin{array}{l} \begin{array}{l} \begin{array}{l} \begin{array}{l} \text{ constraint} \\ \end{array} \\ \hline \begin{array}{l} P \  \  & \text{ bis } \  \  & \text{ bis } \  & \text{ ad wait} \end{array} \\ \hline \begin{array}{l} P \  \  & \text{ bis } \  & \text{ line } \  & \text{ line } \  & \text{ line } \  & \text{ controller} \end{array} \\ \hline \begin{array}{l} \begin{array}{l} \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \\ \hline \begin{array}{l} \begin{array}{l} P \  & \text{ constraint} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \end{array} \end{array} \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \end{array} \end{array} \end{array} \end{array} $ \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \end{array} \end{array} \end{array} \end{array}  \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \end{array}  \\ \hline \begin{array}{l} P \  & \text{ constraint} \end{array} \end{array} \end{array} \end{array} \end{array}  \\ \end{array} \end{array} \end{array} \end{array}  \\ \end{array} \end{array}  \\ \end{array}  \\ \end{array}  \\ \end{array}  \\ \begin{array}{l} P \  & \text{ constraint} \end{array}	$\begin{array}{llllllllllllllllllllllllllllllllllll$	$ \begin{array}{llllllllllllllllllllllllllllllllllll$



Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
<ol> <li>Read page-sized data D<sup>1</sup><sub>2</sub> from NAND array b (: channel controller index b f: NAND flash page index b k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>1</sup><sub>2</sub> from NAND array</li> <li>i: channel controller index</li> <li>j: NAND flash page index</li> <li>k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>f</sup><sub>2</sub> from NAND array</li> <li>▷ i; channel controller index</li> <li>▷ f: NAND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ol>
2: Pull $\theta_{cache}$ from the cache controller buffer	2: Pull $\theta_{\text{cache}}$ from the cache controller buffer	2: $k \leftarrow 0$
3: $\theta^i \leftarrow \theta_{cache}$	3: 0 <sup>+</sup> ← 0 <sub>eacho</sub>	3: while $k < b$ do $\Rightarrow b$ : minibatch size
4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: Calculate $F(D_{ik}^i, \theta^i)$
5: while $k < b$ do $b$ : minibatch size	5: while $k < b$ do $b$ : minibatch size	5: temp $\leftarrow$ temp $+ \eta \nabla F(D_{jk}^{i}, \theta^{i})$
6: Calculate $F(D_{i,i}^i, \theta^i)$	6: Calculate $F(D^i, \theta^i)$	6: $k \leftarrow k+1$
7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{ik}, \theta^i)$	7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{+}, \theta^i)$	7: end while $\mathbf{S}: \mathbf{A}^{\dagger} = \mathbf{A}^{\dagger} = 1$ torus
8: $k \leftarrow k+1$	8: $k \leftarrow k+1$	9: if $j \mod \tau = 0$ then p mod: modulus
9: end while	9: end while	10: Pull $\theta_{cache}$ from the cache controller buffer
10: Push $\Delta \theta^i$ and wait	10: if $\pm \mod \tau = 0$ then $\Rightarrow \mod $ modulos	11: temp $\leftarrow \theta_{cache}$
> Lines 11-12: executed by the cache controller	11: Pash Δθ <sup>4</sup>	12: $\Delta \theta^i \leftarrow \alpha(\theta^i - \text{temp})$
11: $\theta_{\text{curb}\alpha} \leftarrow \theta_{\text{curb}\alpha} - \frac{1}{2} \sum_{i} \Delta \theta^{i}$	12: $[0, \dots, 0] = A[0]$ is by each and	13: $\theta^{i} \leftarrow \theta^{i} - \Delta \theta^{i}$
12: [Signal each channel controller]	13: and if	14: Push $\Delta \theta'$
and a start that the start to be started	1.5. end it	15. (Ceache + Coche + LAO ) > by cache cun







MSST 2017

Algorithm 1 ISP-Based Synchro, SGD Algorithm 3 ISP-Based EASGD Algorithm 2 ISP-Based Downpour SGD 1: Read page-sized data D1 from NAND array 1: Read page-sized data D1 from NAND array 1: Read page-sized data D1 from NAND array b i: channel controller index > i: channel controller index p il channel controller index > j: NAND flash page index > j: NAND flash page index > j: NAND flash page index > k: training sample index (within a minibatch) b k: training sample index (within a minibatch) > k; training sample index (within a minibatch) 2: Pull  $\theta_{carbo}$  from the cache controller buffer 2: Pull  $\theta_{eacher}$  from the cache controller buffer 2:  $k \leftarrow 0$ 3:  $\theta^i \leftarrow \theta_{cuche}$ 3: 0° ← 0 eacho 3: while k < b do p b; minibatch size 4:  $\Delta \theta^i \leftarrow 0, k \leftarrow 0$ 4:  $\Delta \theta^{i} \leftarrow 0, k \leftarrow 0$ 4: Calculate  $F(D^i_{ik}, \theta^i)$ temp  $\leftarrow$  temp  $+ \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 5: while k < b do a be minibatch size 5: while k < b do b b: minibatch size 6:  $k \leftarrow k+1$ Calculate  $F(D_{ik}^{i}, \theta^{i})$ Calculate F(D', , 01) 6: 6: 7: end while  $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{+*}, \theta^i)$ 7:  $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 7: 8:  $\theta^{i} \leftarrow \theta^{i} - l temp$ 8.  $k \leftarrow k+1$ 8:  $k \leftarrow k+1$ 9: if  $j \mod \tau = 0$  then 9: end while 10: Pull  $\theta_{cache}$  from the cache controller buffer Q: end while 10: Push  $\Delta \theta^s$  and wait 11: temp  $\leftarrow \theta_{cache}$ 10: if  $i \mod \tau = 0$  then > mod: modulus  $\Delta \theta^{i} \leftarrow \alpha(\theta^{i} - \text{temp})$ 12. > Lines 11-12: executed by the cache controller 11: Push  $\Delta \theta^{i}$ 13:  $\theta^{i} \leftarrow \theta^{i} - \Delta \theta^{i}$ 11:  $\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{2} \sum_{i} \Delta \theta^{i}$  $\theta_{cache } \leftarrow \theta_{cache } - \Delta \theta^{*}$ 5 by cache ctrl. 14: Push  $\Delta \theta^{\dagger}$ 12: Signal each channel controller 13: end if 15:  $\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} + \Delta \theta^{1}$ 16: end if



> mod: modulus

> by cache ctrl.

Algorithm 1 ISP-Based Synchro, SGD Algorithm 3 ISP-Based EASGD Algorithm 2 ISP-Based Downpour SGD 1: Read page-sized data D1 from NAND array 1: Read page-sized data D1 from NAND array 1: Read page-sized data D1 from NAND array b i: channel controller index > i: channel controller index p il channel controller index > j: NAND flash page index > j: NAND flash page index > j: NAND flash page index > k: training sample index (within a minibatch) b k: training sample index (within a minibatch) > k; training sample index (within a minibatch) 2: Pull  $\theta_{carbo}$  from the cache controller buffer 2: Pull  $\theta_{eacher}$  from the cache controller buffer 2:  $k \leftarrow 0$ 3:  $\theta^i \leftarrow \theta_{cuche}$ 3: 0° ← 0 eacho 3: while k < b do p b; minibatch size 4:  $\Delta \theta^i \leftarrow 0, k \leftarrow 0$ 4:  $\Delta \theta^{i} \leftarrow 0, k \leftarrow 0$ 4: Calculate  $F(D^i_{ik}, \theta^i)$ temp  $\leftarrow$  temp  $+ \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 5: while k < b do a be minibatch size 5: while k < b do b b: minibatch size 6:  $k \leftarrow k+1$ Calculate  $F(D_{ik}^i, \theta^i)$ Calculate F(Dik, 0) 6: 6: 7: end while  $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{+*}, \theta^i)$ 7: 7:  $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 8:  $\theta^{i} \leftarrow \theta^{i} - l temp$ 8.  $k \leftarrow k+1$ 8:  $k \leftarrow k + 1$ 9: if  $j \mod \tau = 0$  then > mod: modulus 9: end while 10: Pull  $\theta_{cache}$  from the cache controller buffer Q. end while 11: 10: Push  $\Delta \theta^s$  and wait temp  $\leftarrow \theta_{cache}$ 10: if  $\pm \mod \tau = 0$  then > mod: modulus  $\Delta \theta^{i} \leftarrow \alpha(\theta^{i} - \text{temp})$ 12: > Lines 11-12: executed by the cache controller Push  $\Delta \theta^{*}$ 13:  $\theta^{i} \leftarrow \theta^{i} - \Delta \theta^{i}$ 11:  $\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{2} \sum_{i} \Delta \theta^{i}$  $\theta_{caches} \leftarrow \theta_{caches} - \Delta \theta^{i}$ b by cache ctrl. 14: Push  $\Delta \theta^{\dagger}$ 12: Signal each channel controller 13: end if  $\theta_{cache} \leftarrow \theta_{cache} + \Delta \theta^{1}$ > by cache ctrl. 16: end if



Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
<ol> <li>Read page-sized data D<sup>1</sup><sub>j</sub> from NAND array</li> <li>▷ i: channel controller index</li> <li>▷ j: NAND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>1</sup><sub>j</sub> from NAND array</li> <li>▷ i: channel controller index</li> <li>▷ j: NAND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>i</sup><sub>j</sub> from NAND array</li> <li>▷ i: channel controller index</li> <li>▷ j: NAND flash page index</li> <li>▷ k.ND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ol>
2: Pull $\theta_{cache}$ from the cache controller buffer	2: Pull $\theta_{cache}$ from the cache controller buffer	2: $k \leftarrow 0$
3: $\theta^i \leftarrow \theta_{cache}$	3: $\theta^* \leftarrow \theta_{encho}$ .	3: while $k < b$ do $\Rightarrow b$ : minibatch size
4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: $\Delta \theta' \leftarrow 0,  k \leftarrow 0$	4: Calculate $F(D_{jk}^i, \theta^i)$
5: while $k < b$ do $b$ : minibatch size	5: while $k < b$ do $b$ : minibatch size	5: temp $\leftarrow$ temp $+ \eta \nabla F(D_{jk}^{i}, \theta^{i})$
6: Calculate $F(D^i_{i,i}, \theta^i)$	6: Calculate $F(D^i, \theta^i)$	6: $k \leftarrow k+1$
7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{jk}, \theta^i)$	7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{ik}, \theta^i)$	7: end while 8: $\theta^{i} \leftarrow \theta^{i} - \frac{1}{2}$ temp
8: $k \leftarrow k+1$	8: $k \leftarrow k+1$	9: if $j \mod \tau = 0$ then $raise \mod 1$
9: end while	9: end while	10: Pull $\theta_{cache}$ from the cache controller buffer
10: Push $\Delta \theta^{i}$ and wait	10: if $j \mod \tau = 0$ then $\Rightarrow \mod: \mod$	11: temp $\leftarrow \theta_{cache}$
> Lines 11-12: executed by the cache controller	11: Pash $\Delta \theta^{i}$	12: $\Delta \theta^* \leftarrow \alpha(\theta^* - \text{temp})$
11: $\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{n} \sum_{i} \Delta \theta^{i}$	12: $\theta_{\text{cache } t} = \theta_{\text{cache }} = \Delta \theta^{t}$ is by cache ctrl.	13: $\theta' \leftarrow \theta' - \Delta \theta'$
12: Signal each channel controller	13: end if	15: $\theta_{eache} \leftarrow \theta_{eache} + \Delta \theta^{1}$ > by cache ctrl. 16: end if



Alexalthes 1 ICD Deced Country CCD	Alexaldhan 2 ICD Based Damasan SCD	Algorithm 2 ICD David EACCD
Algorium 1 ISP-Based Synchro. SOD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 5 ISP-Based EASOD
1: Read page-sized data Dj from NAND array	1: Read page-sized data $D_j^1$ from NAND array	1: Read page-sized data D <sub>j</sub> <sup>i</sup> from NAND array
<ul> <li>▷ i: channel controller index</li> <li>▷ j: NAND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ul>	<ul> <li>i: channel controller index</li> <li>j: NAND flash page index</li> <li>k: training sample index (within a minibatch)</li> </ul>	<ul> <li>i: channel controller index</li> <li>j: NAND flash page index</li> <li>k: training sample index (within a minibatch)</li> </ul>
2: Pull $\theta_{cache}$ from the cache controller buffer	2: Pull $\theta_{\text{cache}}$ from the cache controller buffer	2: $k \leftarrow 0$
3: $\theta^i \leftarrow \theta_{cache}$	3: 0 <sup>+</sup> ← 0 <sub>eacho</sub>	3: while $k < b$ do $\Rightarrow b$ : minibatch size
4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: Calculate $F(D_{jk}^{i}, \theta^{i})$
5: while $k < b$ do $b$ : minibatch size	5: while $k < b$ do $b$ : minibatch size	5: temp $\leftarrow$ temp $+ \eta \nabla F(D_{jk}^{i}, \theta^{i})$
6: Calculate $F(D_{ik}^i, \theta^i)$	6: Calculate $F(D_{ik}^i, \theta^i)$	6: $k \leftarrow k + 1$
7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{jk}, \theta^i)$	7: $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D^{i}_{ik}, \theta^{i})$	8: $\theta^{i} \leftarrow \theta^{i} - \frac{1}{2}$ temp
8: $k \leftarrow k+1$	8: $k \leftarrow k + 1$	9: if $j \mod \tau = 0$ then $\triangleright \mod t$
9: end while	9: end while	10: Pull $\theta_{cache}$ from the cache controller buffer
10: Push $\Delta \theta^i$ and wait	10: If $\pm \mod \tau = 0$ then $\Rightarrow \mod c$	11: temp $\leftarrow \theta_{cache}$
> Lines 11-12: executed by the cache controller	11: Park Ad <sup>1</sup>	12: $\Delta \theta^i \leftarrow \alpha(\theta^i - \text{temp})$
11. 0 0		13: $\theta^i \leftarrow \theta^i - \Delta \theta^i$
12. Pouche + Deache - n Zi 20	12: $\theta_{cache} \leftarrow \theta_{cache} - \Delta \theta'$ to by cache cut.	14: Push $\Delta \theta'$
12. Signal each channel controller	13: end if	15: $\left[\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} + \Delta \theta^*\right] > \text{by cache ctrl.}$
12: Signal each channel controller	13: end if	15: $\theta_{cache} \leftarrow \theta_{cache} + \Delta \theta^{4}$ > by ca 16: end if



Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
<ol> <li>Read page-sized data D<sup>1</sup><sub>j</sub> from NAND array</li> <li>b i: channel controller index</li> <li>b j: NAND flash page index</li> <li>b k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Rend page-sized data D<sup>1</sup><sub>2</sub> from NAND array</li> <li>▷ f: channel controller index</li> <li>▷ f: NAND flash page index</li> <li>▷ f: NAND flash page index</li> <li>▷ f: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>f</sup><sub>2</sub> from NAND array</li> <li>▷ i; channel controller index</li> <li>▷ f: NAND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ol>
2: Pull $\theta_{cache}$ from the cache controller buffer	2: Pull $\theta_{\text{cache}}$ from the cache controller buffer	2: $k \leftarrow 0$
3: $\theta^i \leftarrow \theta_{cache}$	3: 0' + Oencho	3: while $k < b$ do $\Rightarrow b$ : minibatch size
4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: $\Delta \theta^{i} \leftarrow 0, k \leftarrow 0$	4: Calculate $F(D_{jk}^i, \theta^i)$
5: while $k < b$ do $b$ : minibatch size	5: while $k < b$ do $b$ : minibatch size	5: temp $\leftarrow$ temp $+ \eta \nabla F(D_{jk}^{i}, \theta^{i})$
6: Calculate $F(D_{ik}^{i}, \theta^{i})$	6: Calculate $F(D_{i,k}^{i}, \theta^{i})$	6: $k \leftarrow k+1$
7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{ik}, \theta^i)$	7: $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D_{i+1}^{i}, \theta^{i})$	7: end while $8: \theta^1 \leftarrow \theta^1 = 1$ terms
8: $k \leftarrow k+1$	8: $k \leftarrow k + 1$	9: if $j \mod \tau = 0$ then $\triangleright \mod modulus$
9: end while	9: end while	10: Pull $\theta_{cache}$ from the cache controller buffer
10: Push $\Delta \theta^{s}$ and wait	10: if $j \mod \tau = 0$ then $\Rightarrow \mod: \mod$	11: temp $\leftarrow \theta_{cache}$
> Lines 11-12: executed by the cache controller	11: Push $\Delta \theta^i$	12: $\Delta \theta^{\dagger} \leftarrow \alpha(\theta^{\dagger} - \text{temp})$
11: $\theta_{\text{cachs}} \leftarrow \theta_{\text{cachs}} - \frac{1}{n} \sum_{i} \Delta \theta^{i}$	12: $\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} - \Delta \theta^i$ to by cache ctrl.	13: $\theta \leftarrow \theta = \Delta \theta$ 14: Push $\Delta \theta^{\dagger}$
12: Signal each channel controller	13: end if	15: $\theta_{\text{eacher}} \leftarrow \theta_{\text{eacher}} + \Delta \theta^{1}$ > by cache ctrl.
and the second sec		16: end if



MSST 2017

Algorithm 1 ISP-Based Synchro, SGD Algorithm 3 ISP-Based EASGD Algorithm 2 ISP-Based Downpour SGD 1: Read page-sized data D1 from NAND array 1: Read page-sized data D1 from NAND array 1: Read page-sized data D1 from NAND array b i: channel controller index > i: channel controller index p il channel controller index > j: NAND flash page index > j: NAND flash page index > j: NAND flash page index > k: training sample index (within a minibatch) b k: training sample index (within a minibatch) to k; training sample index (within a minibatch) 2: Pull  $\theta_{carbo}$  from the cache controller buffer 2: Pull  $\theta_{eacher}$  from the cache controller buffer 2:  $k \leftarrow 0$ 3:  $\theta^i \leftarrow \theta_{cuche}$ 3: 0' ← 0 eacho 3: while k < b do p b; minibatch size 4:  $\Delta \theta^i \leftarrow 0, k \leftarrow 0$ 4:  $\Delta \theta^{i} \leftarrow 0, k \leftarrow 0$ 4: Calculate  $F(D^i_{ik}, \theta^i)$ temp  $\leftarrow$  temp  $+ \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 5: while k < b do a be minibatch size 5: while k < b do b b: minibatch size 6:  $k \leftarrow k+1$ Calculate  $F(D_{ik}^i, \theta^i)$ Calculate F(D', , 01) 6: 6: 7: end while  $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{+*}, \theta^i)$  $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 7: 7: 8:  $\theta^{i} \leftarrow \theta^{i} - l temp$ 8.  $k \leftarrow k+1$ 8:  $k \leftarrow k + 1$ 9: if  $\tau \mod \tau = 0$  then b mod: mod 9: end while 10: Pull  $\theta_{cache}$  from the cache controller buffer 9: end while 11: temp  $\leftarrow \theta_{cache}$ 10: Push  $\Delta \theta^s$  and wait 10: if  $i \mod \tau = 0$  then > mod: modulus  $\Delta \theta^{i} \leftarrow \alpha(\theta^{i} - \text{temp})$ 12: > Lines 11-12: executed by the cache controller Push  $\Delta \theta^{i}$ HE: 13:  $\theta^{i} \leftarrow \theta^{i} - \Delta \theta^{i}$ 11:  $\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{2} \sum_{i} \Delta \theta^{i}$  $\theta_{cache } \leftarrow \theta_{cache } - \Delta \theta^{*}$ to by cache ctrl. 14: Push  $\Delta \theta^1$ 12: Signal each channel controller 13: end if 15:  $\theta_{cache} \leftarrow \theta_{cache} + \Delta \theta^{1}$ > by cache ctrl. 16: end if



Algorithm 1 ISP-Based Synchro, SGD Algorithm 3 ISP-Based EASGD Algorithm 2 ISP-Based Downpour SGD 1: Read page-sized data D1 from NAND array 1: Read page-sized data D1 from NAND array 1: Read page-sized data D, from NAND array b i: channel controller index > i: channel controller index p il channel controller index > j: NAND flash page index > j: NAND flash page index > j: NAND flash page index > k: training sample index (within a minibatch) b k: training sample index (within a minibatch) > k; training sample index (within a minibatch) 2: Pull  $\theta_{carbo}$  from the cache controller buffer 2: Pull  $\theta_{eacher}$  from the cache controller buffer 2:  $k \leftarrow 0$ 3:  $\theta^i \leftarrow \theta_{cuche}$ 3: 0° ← 0 eacho 3: while k < b do p b; minibatch size 4:  $\Delta \theta^i \leftarrow 0, k \leftarrow 0$ 4:  $\Delta \theta^{i} \leftarrow 0, k \leftarrow 0$ 4: Calculate  $F(D^i_{ik}, \theta^i)$ 5: temp  $\leftarrow$  temp  $+ \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 5: while k < b do a be minibatch size 5: while k < b do b b: minibatch size 6:  $k \leftarrow k+1$ Calculate  $F(D_{ik}^i, \theta^i)$ Calculate F(D'1, 0') 6: 6: 7: end while  $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{+*}, \theta^i)$  $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D_{ik}^{i}, \theta^{i})$ 7: 7: 8:  $\theta^{i} \leftarrow \theta^{i} - l temp$ 8.  $k \leftarrow k+1$ 8:  $k \leftarrow k + 1$ 9: if  $j \mod \tau = 0$  then > mod: modula Pull  $\theta_{cache}$  from the cache controller buffer 9: end while 10: 9: end while 10: Push  $\Delta \theta^s$  and wait 11: temp  $\leftarrow \theta_{cache}$ 10: if  $i \mod \tau = 0$  then > mod: modulus  $\Delta \theta^i \leftarrow \alpha(\theta^i - \text{temp})$ > Lines 11-12: executed by the cache controller H: Push  $\Delta \theta^{i}$ 13:  $\theta^{i} \leftarrow \theta^{i} - \Delta \theta^{i}$ 11:  $\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{2} \sum_{i} \Delta \theta^{i}$  $\theta_{cache } \leftarrow \theta_{cache } - \Delta \theta^{*}$ to by cache ctrl. 14: Push  $\Delta \theta^1$ 12: Signal each channel controller 13: end if 15:  $\theta_{cache} \leftarrow \theta_{cache} + \Delta \theta^{1}$ > by cache ctrl. 16: end if



MSST 2017

Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
<ol> <li>Read page-sized data D<sup>1</sup><sub>2</sub> from NAND array</li> <li>▷ i: channel controller index</li> <li>▷ j: NAND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>1</sup><sub>2</sub> from NAND array</li> <li>i: channel controller index</li> <li>j: NAND flash page index</li> <li>k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>i</sup><sub>j</sub> from NAND array</li> <li>▷ i: channel controller index</li> <li>▷ j: NAND flash page index</li> <li>▷ k: training sample index (within a minibatch)</li> </ol>
2: Pull $\theta_{cache}$ from the cache controller buffer	2: Pull $\theta_{\text{cache}}$ from the cache controller buffer	2: $k \leftarrow 0$
3: $\theta^i \leftarrow \theta_{cache}$	3: $\theta^* \leftarrow \theta_{eacho}$	3: while $k < b$ do $\Rightarrow b$ : minibatch size
4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: Calculate $F(D_{jk}^i, \theta^i)$
5: while $k < b$ do $b$ : minibatch size	5: while $k < b$ do $b$ : minibatch size	5: temp $\leftarrow$ temp $+ \eta \nabla F(D_{jk}^{i}, \theta^{i})$
6: Calculate $F(D_{ik}^i, \theta^i)$	6: Calculate $F(D_{i,k}^{i}, \theta^{i})$	6: $k \leftarrow k + 1$
7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{1k}, \theta^i)$	7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{i+1}, \theta^i)$	7: end while 8: $\theta^{i} \leftarrow \theta^{i} = \lambda$ terms
8: $k \leftarrow k+1$	8: $k \leftarrow k+1$	9: if $j \mod \tau = 0$ then $raise \mod 1$
9: end while	9: end while	10: Pull $\theta_{cache}$ from the cache controller buffer
10: Push $\Delta \theta^{i}$ and wait	10: if $j \mod \tau = 0$ then $> \mod$ : modulus	11: temp $\leftarrow \theta_{cache}$
> Lines 11-12: executed by the cache controller	11: Push $\Delta \theta^i$	12: $\Delta \theta^{\dagger} \leftarrow \alpha(\theta^{\dagger} - \text{temp})$
11: $\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{n} \sum_{i} \Delta \theta^{i}$	12: $\theta_{eache} \leftarrow \theta_{eache} - \Delta \theta^*$ to by eache ctrl.	13: $\theta' \leftarrow \theta' - \Delta \theta'$
12: Signal each channel controller	13: end if	15: $\theta_{\text{eacher}} \leftarrow \theta_{\text{eacher}} + \Delta \theta^{\text{t}}$ > by cache ctrl.
		16: end if



Algorithm 1 ISP-Based Synchro. SGD	Algorithm 2 ISP-Based Downpour SGD	Algorithm 3 ISP-Based EASGD
<ol> <li>Read page-sized data D<sup>i</sup><sub>j</sub> from NAND array</li> <li>i: channel controller index</li> <li>j: NAND flash page index</li> <li>k: training sample index (within a minibatch)</li> </ol>	<ol> <li>Rend page-sized data D<sup>1</sup><sub>2</sub> from NAND array</li> <li>▷ i: channel controller index</li> <li>▷ j: NAND flash page index</li> <li>▷ i: training sample index (within a minibatch)</li> </ol>	<ol> <li>Read page-sized data D<sup>i</sup><sub>j</sub> from NAND array</li> <li>i: channel controller index</li> <li>j: NAND flash page index</li> <li>k: training sample index (within a minibatch)</li> </ol>
2: Pull $\theta_{cache}$ from the cache controller buffer	2: Pull $\theta_{\text{rache}}$ from the cache controller buffer	2: $k \leftarrow 0$
3: $\theta^i \leftarrow \theta_{cache}$	3: $\theta^* \leftarrow \theta_{eacho}$ .	3: while $k < b$ do $\Rightarrow b$ : minibatch size
4: $\Delta \theta^{i} \leftarrow 0,  k \leftarrow 0$	4: $\Delta \theta^i \leftarrow 0,  k \leftarrow 0$	4: Calculate $F(D_{jk}^i, \theta^i)$
5: while $k < b$ do $b$ : minibatch size	5: while $k < b$ do $b$ : minibatch size	5: temp $\leftarrow$ temp $+ \eta \nabla F(D_{jk}^*, \theta^*)$
6: Calculate $F(D_{ik}^i, \theta^i)$	6: Calculate $F(D_{ik}^{i}, \theta^{i})$	$b: k \leftarrow k + 1$
7: $\Delta \theta^i \leftarrow \Delta \theta^i + \eta \nabla F(D^i_{jk}, \theta^i)$	7: $\Delta \theta^{i} \leftarrow \Delta \theta^{i} + \eta \nabla F(D^{i}_{ik}, \theta^{i})$	8: $\theta^{i} \leftarrow \theta^{i} - \frac{1}{2}$ temp
8: $k \leftarrow k+1$	8: $k \leftarrow k + 1$	9: if $j \mod \tau = 0$ then $\triangleright \mod t$ modulas
9: end while	9: end while	10: Pull $\theta_{cache}$ from the cache controller buffer
10: Push $\Delta \theta^i$ and wait	10: if $j \mod \tau = 0$ then $\Rightarrow \mod: \mod$	11: temp $\leftarrow \theta_{cache}$
> Lines 11-12: executed by the cache controller	11: Push $\Delta \theta^{i}$	12: $\Delta \theta^{\dagger} \leftarrow \alpha(\theta^{\dagger} - \text{temp})$
11: $\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{n} \sum_{i} \Delta \theta^{i}$	12: $\theta_{exche} \leftarrow \theta_{exche} - \Delta \theta^{*}$ is by cache ctrl.	13: $\theta' \leftarrow \theta' - \Delta \theta'$
12: Signal each channel controller	13: end if	15: $\theta_{-++} \neq \theta_{-++} + \Delta \theta^{+}$ b by cache stri
		Ib: end if



# Methodology for IHP-ISP Performance Comparison

- Ideal Ways to Fairly Compare ISP and IHP
  - **(**) Implementing ISP-ML in a real semiconductor chip
    - High chip manufacturing costs
  - **2** Simulating IHP in the ISP-ML framework.
    - High simulation time to simulate IHP
  - **③** Implementing both ISP and IHP using FPGAs.
    - Require another significant development efforts.
- $\Rightarrow$  Hard to fairly compare the performances of ISP and IHP
- $\Rightarrow$  We propose a practical comparison methodology

# Methodology for IHP-ISP Performance Comparison



- Observed IHP execution time  $= T_{total} = T_{nonIO} + T_{IO}$ .
- Expected IHP simulation time =  $T_{nonIO} + T_{IOsim}$ =  $T_{total} - T_{IO} + T_{IOsim}$ .

#### $T_{IO}$ : Data IO latency time of the storage

- $T_{nonIO}$ : Non-data IO time
- $T_{IOsim}\;$  : Data IO time of the baseline SSD in ISP-ML

Hyeokjun Choe et al

MSST 2017

# Outline

#### 1 Introduction

- 2 Background
- B Proposed Methodology
- 4 Experimental Results
  - 5 Discussion and Conclusion

# Setup and Implementation

#### • Host specifications

CPU	8-core Intel(R) Core i7-3770K $(3.50GHz)$
Main memory	DDR3 32GB RAM
Storage	Samsung SSD 840 Pro
OS	Ubuntu 14.04 LTS

#### • ISP-ML specifications

Embedded processor	ARM 926EJ-S (400MHz)
FTL	$\mathrm{DFTL}$
Page size	8KB
$t_{prog} \ / \ t_{read} \ / \ t_{blockerase}$	300us / 75us / 5ms
FPU	0.5 instruction/cycle(pipelined)
Dataset	x10 amplified MNIST(handwritten digits)

# Performance Comparison: ISP-Based Optimization

• EASGD showed best performance in this experiment.

- x2.96 against synchornous SGD on average.
- x1.41 against Downpour SGD on average.
- For 4,8 Ch, synchronous SGD was slower than Downpour SGD
- For 16 Ch, synchronous SGD was faster than Downpour SGD



# Performance Comparison:IHP versus ISP

- Compared IHP in memory shortage situation with ISP
  - In large-scale machine learning, the computing systems used may suffer from memory shortages.
  - Assumption: The host had already loaded all the data to main memory for IHP.
- ISP-based EASGD with 16 channels obtained the best performance in our experiments.



# Channel Parallelism

- The speed-up tends to be proportional to the number of channels.
- Because the communication overhead in ISP is negligible.
  - In distributed computing systems, communication bottleneck commonly occurs.



Hyeokjun Choe et al

MSST 2017

# Effects of Communication Period in Async. SGD

- Downpour SGD
  - High speed for a low communication period  $[\tau{=}1;\,4]$
  - Unstable for a high communication period  $[\tau=16; 64]$
- EASGD
  - Communication period  $\Uparrow,$  convergence speed  $\Downarrow$
  - In contrast to the distributed computing system
  - Because of the low communication overhead



- EASGD shows the best performance in our ISP-ML environment.
- ISP is more efficient than IHP while host suffers from insufficient main memory.
  - ISP may be useful in large scale machine learning.
- The speed-up by parallelizing is proportional to the number of channels.
  - Because of the ultra fast on-chip communication.
- The performance of EASGD decreases while the communication period increases unlike conventional distributed system.

# Outline

#### 1 Introduction

- 2 Background
- 3 Proposed Methodology
- 4 Experimental Results
- **5** Discussion and Conclusion

- ISP can provide various advantages for data processing involved in machine learning.
  - E.g. ultra-fast on-chip communication

 $\Rightarrow$  Increase energy efficiency, security, and reliability

- High degree of parallelism could be achieved.
  - By increasing the number of channels inside an SSD.
- Exploiting a hierarchy of parallelism
  - $\bullet\,$  Distributed systems + ISP-based SSDs

- **()** Implementing deep neural networks in ISP-ML framework
- **2** Implementing adaptive optimization algorithms
  - E.g. Adagrad and Adadelta
- In Pre-computing metadata during data writes
- Implementing data shuffling functionality
- **③** Investigate the effect of NAND flash design on performance

- Create full-fledged ISP-supporting SSD simulator supporting ML
- Implement and compare multiple versions of parallel SGD
- Propose fair comparison methodology between IHP and ISP
- Intrigue future research opportunities in terms of exploiting the channel parallelism

#### Acknowledgments





Ministry of Science, ICT and Future Planning







Hyeokjun Choe et al.

MSST 2017

May 19th, 2017 32 / 33

