Near-Data Processing for Differentiable Machine Learning Models

Hyeokjun Choe[1], Seil Lee[1], Hyunha Nam[1], Seongsik Park[1], Seijoon Kim[1], Eui-Young Chung[2] and Sungroh Yoon[1,3*]

[1]Electrical and Computer Engineering, Seoul National University

[2]Electrical and Electronic Engineering, Yonsei University

[3]Neurology and Neurological Sciences, Stanford University

[*]Correspondence: sryoon@snu.ac.kr

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

May 19th, 2017

Outline

Outline

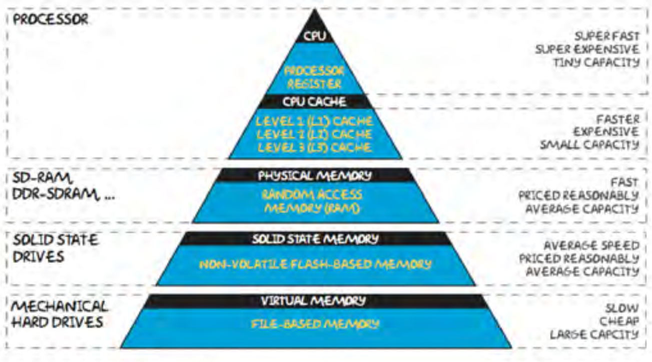
Machine Learning's Success

- Big data
- Powerful parallel processors

⇒ Sophisticated models

Issues on Conventional Memory Hierachy

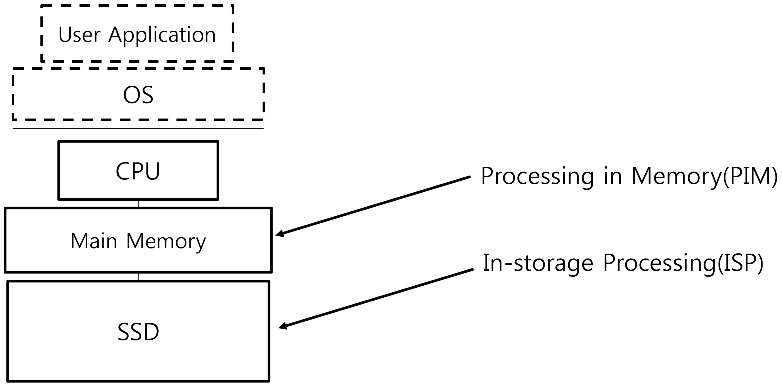
- Data movement in memory hierarchy
  - Computational efficiency ⇓
  - Power consumption ⇑



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

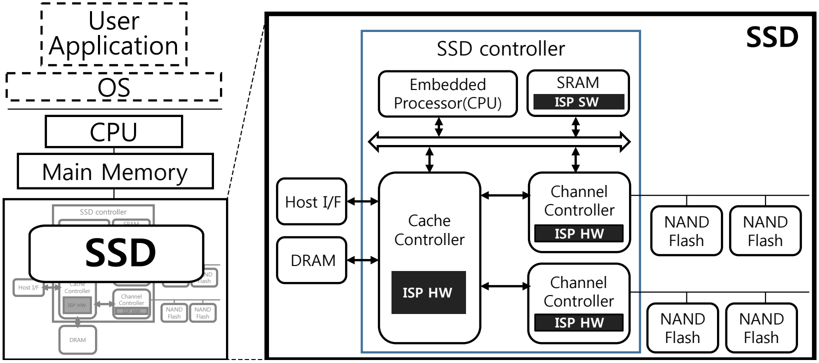
Near-data Processing (NDP)

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



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

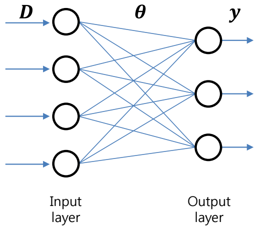


Outline

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 $F(D;\boldsymbol{\theta})$

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

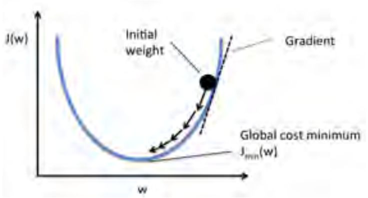


D	:	input data
$\boldsymbol{\theta}$	:	model parameters
L	:	loss function
r	:	regularization term
F	:	objective function

Gradient Descent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla F(D, \boldsymbol{\theta}_t) \quad (2)$$

$$= \boldsymbol{\theta}_t - \eta \sum_i \nabla F(D_i, \boldsymbol{\theta}_t) \quad (3)$$



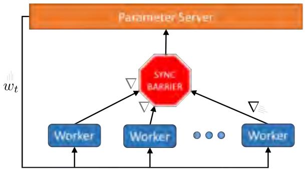
https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html

η	: learning rate
t	: iteration index
i	: data sample index

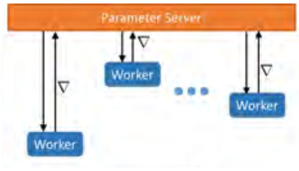
- 1st-order iterative optimization algorithm
  - Use all samples per iteration
- Stochastic gradient descent (SGD)
  - Use only one sample per iteration.
- Minibatch stochastic gradient descent
  - Between gradient descent and SGD
  - Use multiple samples per iteration

Parallel and Distributed SGD

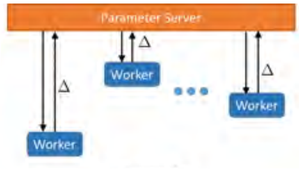
- Synchornous SGD
  - Parameter server aggregates $\nabla\boldsymbol{\theta}_{slave}$ synchronously.
- Downpour SGD
  - Workers communicate with parameter server asynchronously.
- Elastic Average SGD (EASGD)
  - Each worker has own parameters
  - Workers transfer $(\boldsymbol{\theta}_{slave} - \boldsymbol{\theta}_{master})$,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



(a) Synchronous SGD



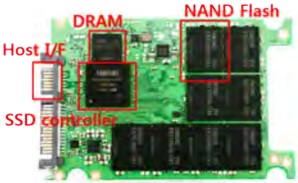
(b) Downpur SGD



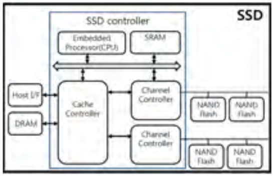
(c) EASGD

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

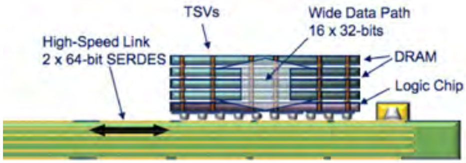


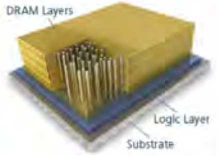
Source: http://www.storagereview.com/samsung_ssd_840_review_tlc



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

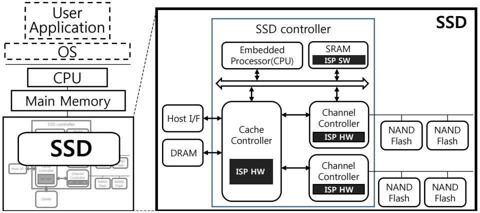
Previous Work on Near-Data Processing:ISP

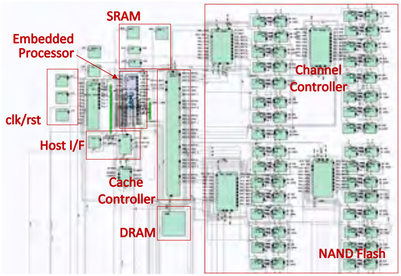
- Perform computation inside the storage
- ISP with embedded processor
  - Pros: easy to implement, flexible
  - Cons: no parallelism
- ISP with dedicated hardware logic
  - Pros: channel parallelism, hardware acceleration
  - Cons: hard to implement and change
- Applications: DB query (scan, join),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

Outline

ISP-ML: ISP Platform for Machine Learning on SSDs

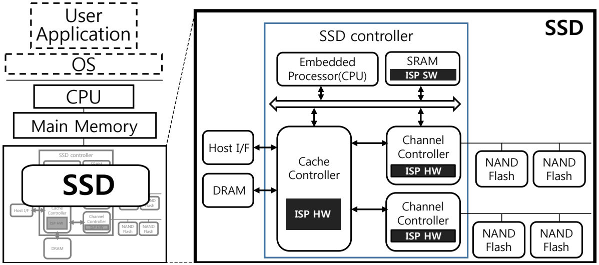
- ISP-supporting SSD simulator
  - Implemented in SystemC on the Synopsys Platform Architect
    - Software/Hardware co-simulation
    - Easily executes various machine learning algorithms in C/C++
- Transaction level simulator
  - For reasonable simulation speed
- ISP components
  - ISP SW, ISP HW



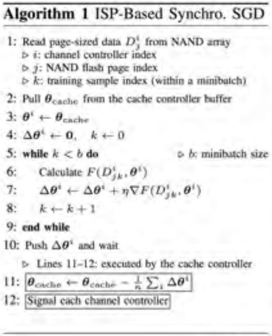


ISP-ML: ISP Platform for Machine Learning on SSDs

- 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.

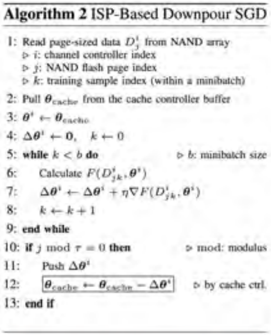


Parallel SGD Implementation on ISP-ML



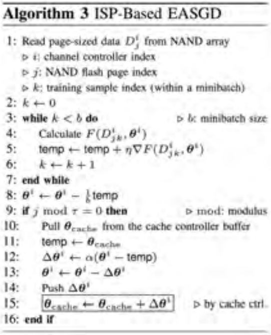
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                                  ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



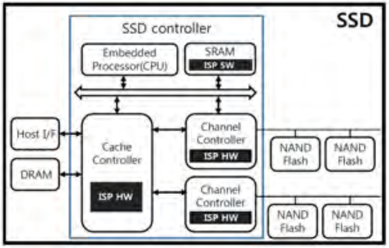
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                                  ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then                        ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
13: end if
```

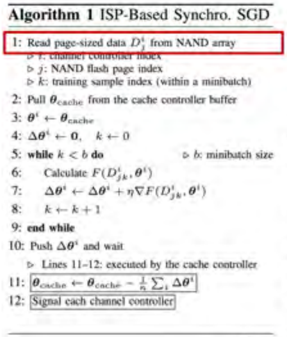


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                                  ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
5:     $\text{temp} \leftarrow \text{temp} + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{b}\text{temp}$
9: if $j \bmod \tau = 0$ then                         ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     $\text{temp} \leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i - \text{temp})$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
16: end if
```

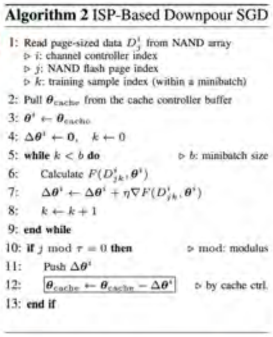


Parallel SGD Implementation on ISP-ML



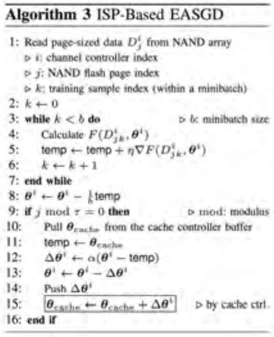
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D^i_j$ from NAND array
       ▷ $i$: channel controller index
       ▷ $j$: NAND flash page index
       ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\mathrm{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\mathrm{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
       ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\mathrm{cache}} \leftarrow \boldsymbol{\theta}_{\mathrm{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: Signal each channel controller
```



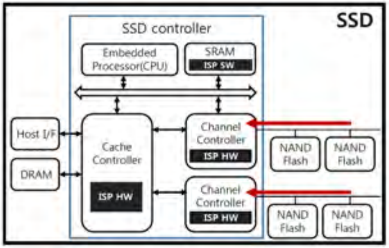
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D^i_j$ from NAND array
       ▷ $i$: channel controller index
       ▷ $j$: NAND flash page index
       ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\mathrm{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\mathrm{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j$ mod $\tau = 0$ then                ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\mathrm{cache}} \leftarrow \boldsymbol{\theta}_{\mathrm{cache}} - \Delta\boldsymbol{\theta}^i}$       ▷ by cache ctrl.
13: end if
```

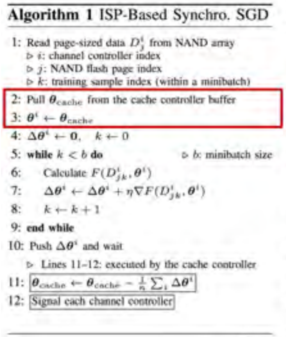


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
       ▷ $i$: channel controller index
       ▷ $j$: NAND flash page index
       ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                         ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{b}$temp
9: if $j$ mod $\tau = 0$ then                 ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\mathrm{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\mathrm{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i -$ temp$)$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\mathrm{cache}} \leftarrow \boldsymbol{\theta}_{\mathrm{cache}} + \Delta\boldsymbol{\theta}^i}$       ▷ by cache ctrl.
16: end if
```

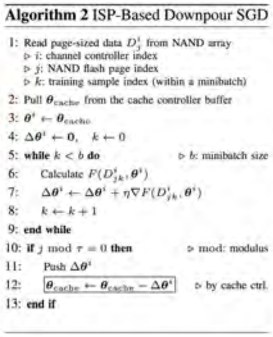


Parallel SGD Implementation on ISP-ML



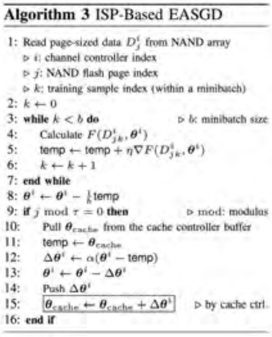
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



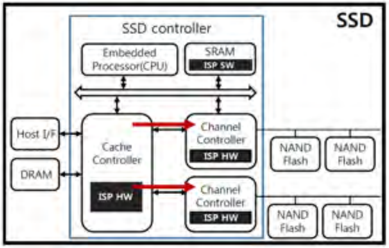
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then               ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$      ▷ by cache ctrl.
13: end if
```

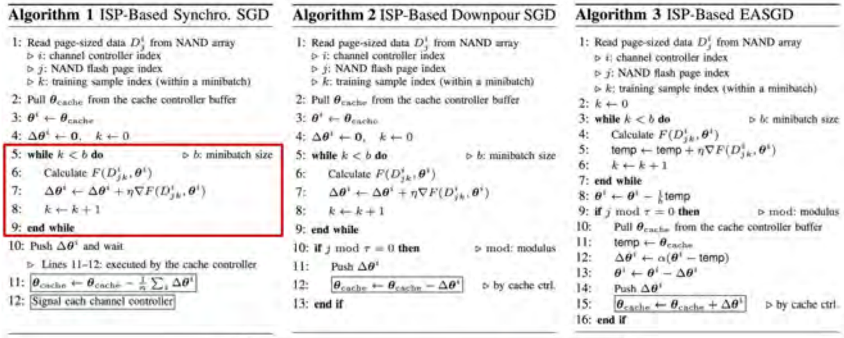


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                          ▷ $b$: minibatch size
4:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{b}$temp
9: if $j \bmod \tau = 0$ then                ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i - \text{temp})$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$      ▷ by cache ctrl.
16: end if
```



Parallel SGD Implementation on ISP-ML



Algorithm 1 ISP-Based Synchro. SGD

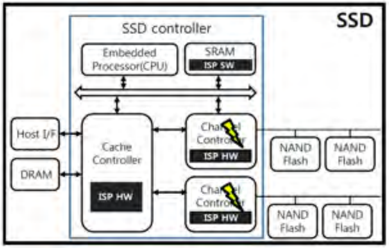
```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                                ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```

Algorithm 2 ISP-Based Downpour SGD

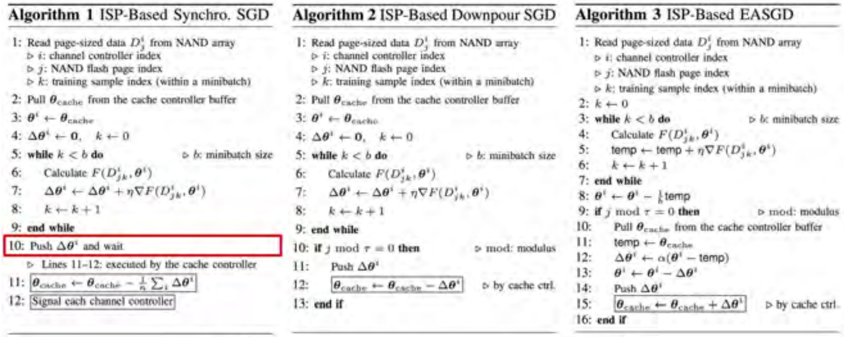
```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                                ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then                      ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$        ▷ by cache ctrl.
13: end if
```

Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                                ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
5:     $\text{temp} \leftarrow \text{temp} + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{b}\text{temp}$
9: if $j \bmod \tau = 0$ then                       ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     $\text{temp} \leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i - \text{temp})$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$        ▷ by cache ctrl.
16: end if
```



Parallel SGD Implementation on ISP-ML



Algorithm 1 ISP-Based Synchro. SGD

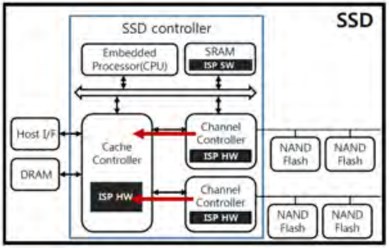
```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```

Algorithm 2 ISP-Based Downpour SGD

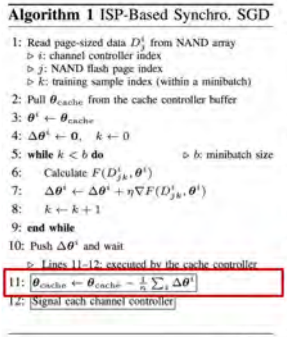
```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j$ mod $\tau = 0$ then                ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$       ▷ by cache ctrl.
13: end if
```

Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                          ▷ $b$: minibatch size
4:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{k}$temp
9: if $j$ mod $\tau = 0$ then                 ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i - \text{temp})$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$       ▷ by cache ctrl.
16: end if
```

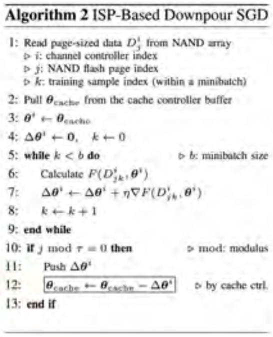


Parallel SGD Implementation on ISP-ML



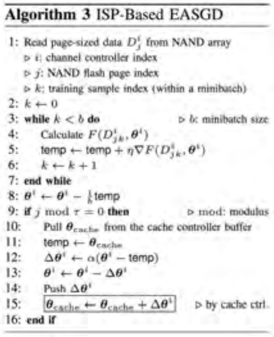
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                              ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



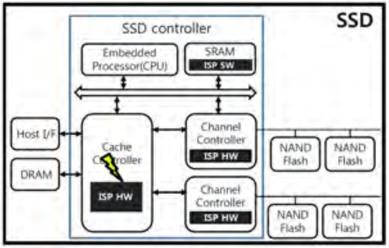
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                              ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then                  ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
13: end if
```

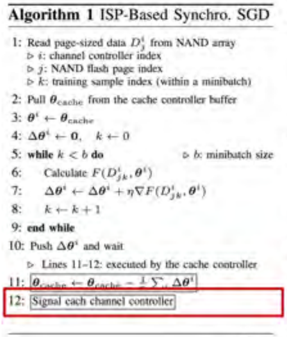


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                              ▷ $b$: minibatch size
4:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{b}$temp
9: if $j \bmod \tau = 0$ then                   ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i - \text{temp})$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
16: end if
```

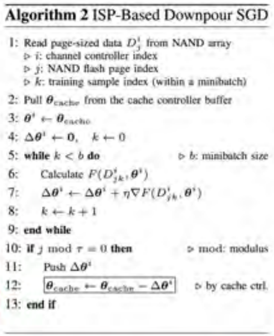


Parallel SGD Implementation on ISP-ML



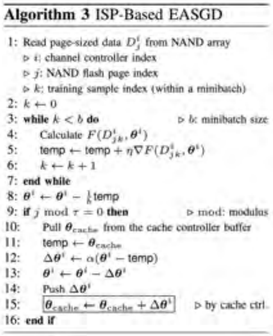
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: Signal each channel controller
```



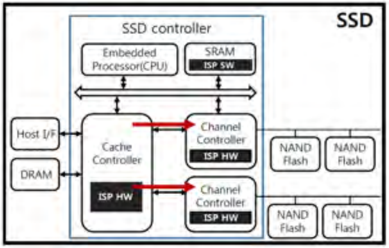
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then                 ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
13: end if
```

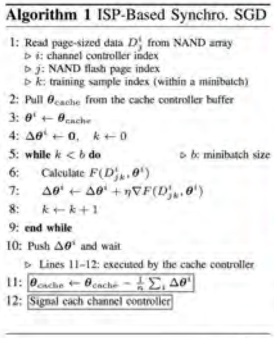


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                          ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{b}$temp
9: if $j \bmod \tau = 0$ then                  ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i -$ temp$)$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
16: end if
```

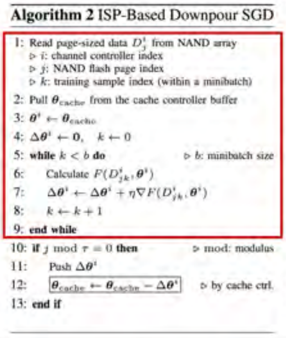


Parallel SGD Implementation on ISP-ML



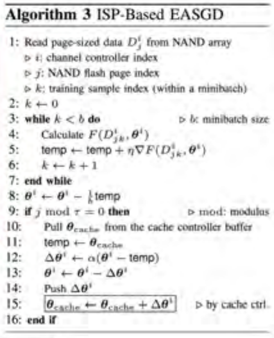
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{\text{cache}}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{\text{cache}}$
4: $\Delta\theta^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D^i_{jk}, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\theta^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} - \frac{1}{n}\sum_i \Delta\theta^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



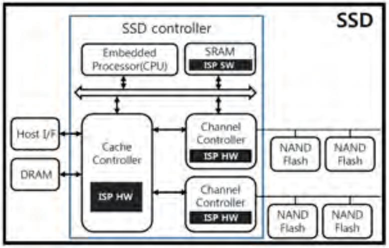
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{\text{cache}}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{\text{cache}}$
4: $\Delta\theta^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D^i_{jk}, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j$ mod $\tau = 0$ then                 ▷ mod: modulus
11:     Push $\Delta\theta^i$
12:     $\boxed{\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} - \Delta\theta^i}$     ▷ by cache ctrl.
13: end if
```

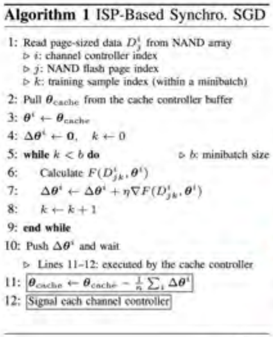


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                          ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \theta^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D^i_{jk}, \theta^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\theta^i \leftarrow \theta^i - \frac{1}{k}$temp
9: if $j$ mod $\tau = 0$ then                  ▷ mod: modulus
10:     Pull $\theta_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \theta_{\text{cache}}$
12:     $\Delta\theta^i \leftarrow \alpha(\theta^i -$ temp$)$
13:     $\theta^i \leftarrow \theta^i - \Delta\theta^i$
14:     Push $\Delta\theta^i$
15:     $\boxed{\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} + \Delta\theta^i}$     ▷ by cache ctrl.
16: end if
```

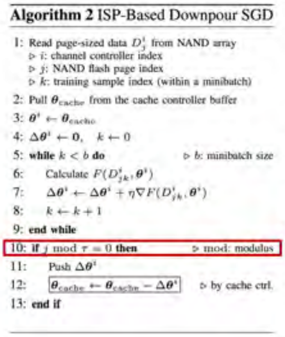


Parallel SGD Implementation on ISP-ML



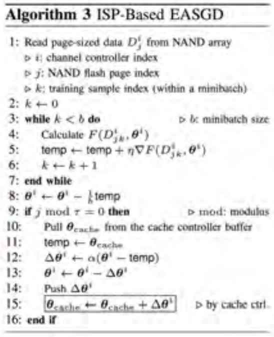
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{cache}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{cache}$
4: $\Delta\theta^i \leftarrow \mathbf{0}$,   $k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D^i_{jk}, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\theta^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{n}\sum_i \Delta\theta^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



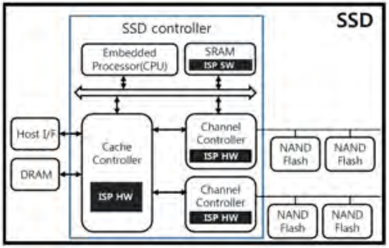
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{cache}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{cache}$
4: $\Delta\theta^i \leftarrow \mathbf{0}$,   $k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D^i_{jk}, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j$ mod $\tau = 0$ then                ▷ mod: modulus
11:     Push $\Delta\theta^i$
12:     $\boxed{\theta_{cache} \leftarrow \theta_{cache} - \Delta\theta^i}$     ▷ by cache ctrl.
13: end if
```

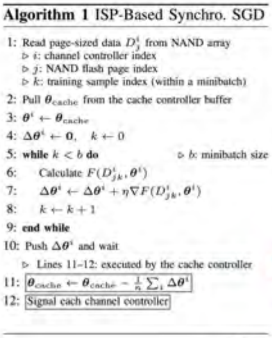


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                         ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \theta^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D^i_{jk}, \theta^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\theta^i \leftarrow \theta^i - \frac{1}{b}$temp
9: if $j$ mod $\tau = 0$ then                 ▷ mod: modulus
10:     Pull $\theta_{cache}$ from the cache controller buffer
11:     temp $\leftarrow \theta_{cache}$
12:     $\Delta\theta^i \leftarrow \alpha(\theta^i - \text{temp})$
13:     $\theta^i \leftarrow \theta^i - \Delta\theta^i$
14:     Push $\Delta\theta^i$
15:     $\boxed{\theta_{cache} \leftarrow \theta_{cache} + \Delta\theta^i}$     ▷ by cache ctrl.
16: end if
```

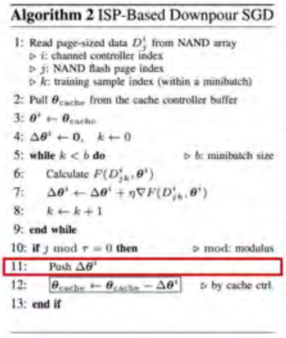


Parallel SGD Implementation on ISP-ML



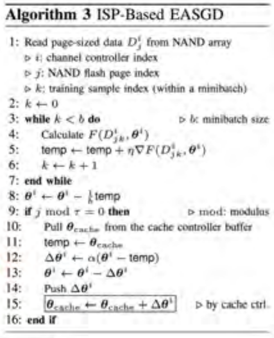
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{cache}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{cache}$
4: $\Delta\theta^i \leftarrow \mathbf{0}$,  $k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D^i_{jk}, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\theta^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{n}\sum_i \Delta\theta^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



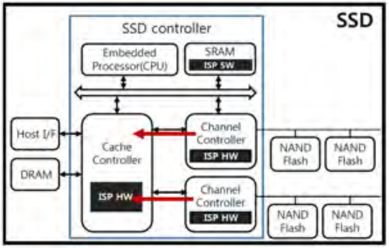
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{cache}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{cache}$
4: $\Delta\theta^i \leftarrow \mathbf{0}$,  $k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D^i_{jk}, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j$ mod $\tau = 0$ then                ▷ mod: modulus
11:     Push $\Delta\theta^i$
12:     $\boxed{\theta_{cache} \leftarrow \theta_{cache} - \Delta\theta^i}$      ▷ by cache ctrl.
13: end if
```

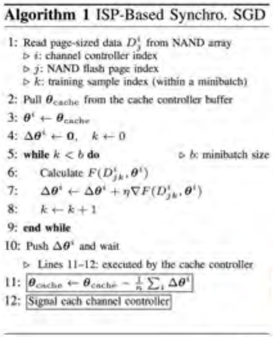


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                         ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \theta^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D^i_{jk}, \theta^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\theta^i \leftarrow \theta^i - \frac{1}{b}$temp
9: if $j$ mod $\tau = 0$ then                 ▷ mod: modulus
10:     Pull $\theta_{cache}$ from the cache controller buffer
11:     temp $\leftarrow \theta_{cache}$
12:     $\Delta\theta^i \leftarrow \alpha(\theta^i - \text{temp})$
13:     $\theta^i \leftarrow \theta^i - \Delta\theta^i$
14:     Push $\Delta\theta^i$
15:     $\boxed{\theta_{cache} \leftarrow \theta_{cache} + \Delta\theta^i}$      ▷ by cache ctrl.
16: end if
```

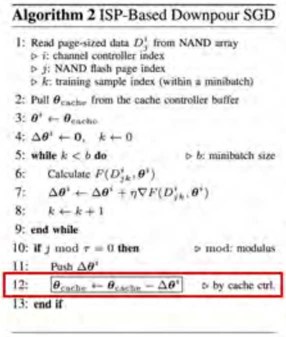


Parallel SGD Implementation on ISP-ML



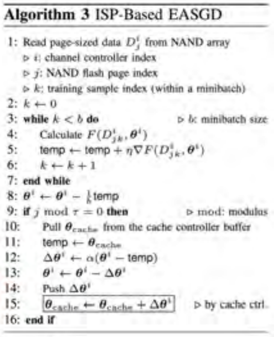
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{\text{cache}}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{\text{cache}}$
4: $\Delta\theta^i \leftarrow \mathbf{0}$,   $k \leftarrow 0$
5: while $k < b$ do                    ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D_{jk}^i, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\theta^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} - \frac{1}{n}\sum_i \Delta\theta^i}$
12: Signal each channel controller
```



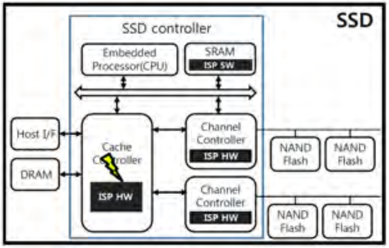
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{\text{cache}}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{\text{cache}}$
4: $\Delta\theta^i \leftarrow \mathbf{0}$,   $k \leftarrow 0$
5: while $k < b$ do                    ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D_{jk}^i, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j$ mod $\tau = 0$ then          ▷ mod: modulus
11:     Push $\Delta\theta^i$
12:     $\boxed{\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} - \Delta\theta^i}$   ▷ by cache ctrl.
13: end if
```

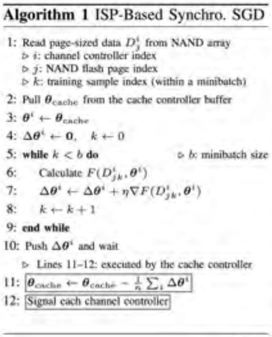


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                    ▷ $b$: 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: $\theta^i \leftarrow \theta^i - \frac{1}{b}$temp
9: if $j$ mod $\tau = 0$ then           ▷ mod: modulus
10:     Pull $\theta_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \theta_{\text{cache}}$
12:     $\Delta\theta^i \leftarrow \alpha(\theta^i - \text{temp})$
13:     $\theta^i \leftarrow \theta^i - \Delta\theta^i$
14:     Push $\Delta\theta^i$
15:     $\boxed{\theta_{\text{cache}} \leftarrow \theta_{\text{cache}} + \Delta\theta^i}$   ▷ by cache ctrl.
16: end if
```

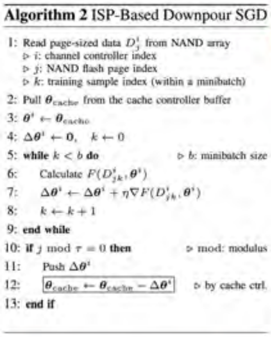


Parallel SGD Implementation on ISP-ML



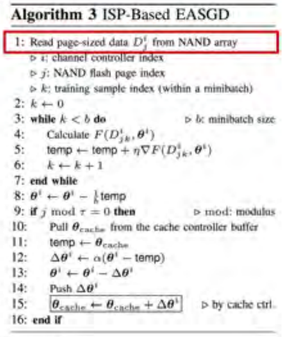
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                                ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



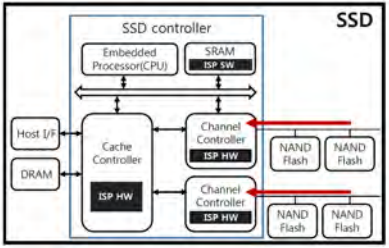
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                                ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then                     ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$      ▷ by cache ctrl.
13: end if
```

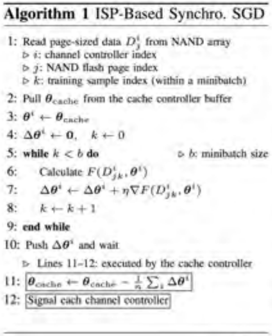


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                                ▷ $b$: minibatch size
4:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
5:     $\text{temp} \leftarrow \text{temp} + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{b}\text{temp}$
9: if $j \bmod \tau = 0$ then                      ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     $\text{temp} \leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i - \text{temp})$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$      ▷ by cache ctrl.
16: end if
```

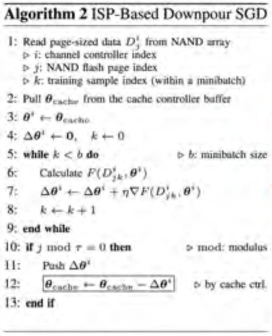


Parallel SGD Implementation on ISP-ML



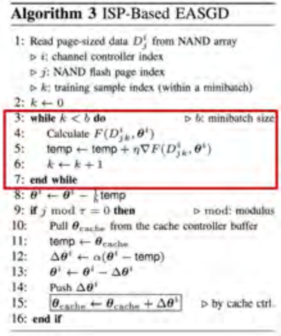
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



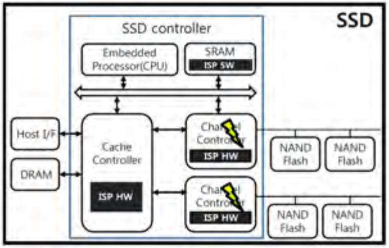
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then                ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
13: end if
```

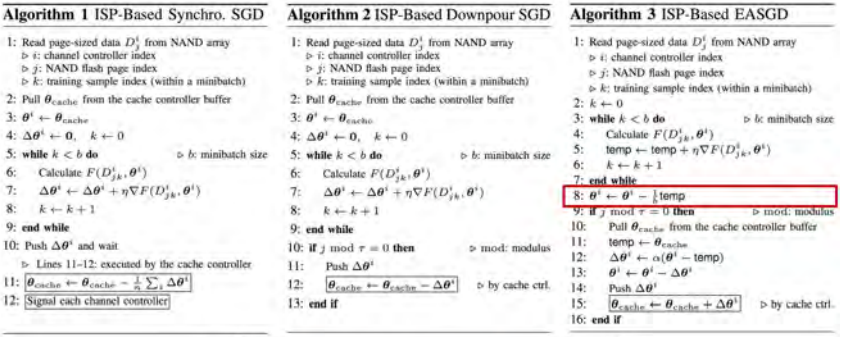


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                         ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{k}$temp
9: if $j \bmod \tau = 0$ then                 ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i -$ temp$)$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
16: end if
```



Parallel SGD Implementation on ISP-ML



Algorithm 1 ISP-Based Synchro. SGD

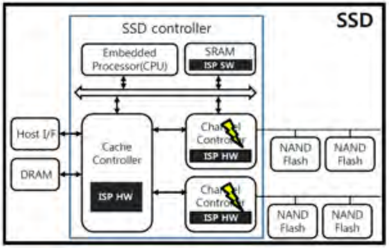
```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                        ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```

Algorithm 2 ISP-Based Downpour SGD

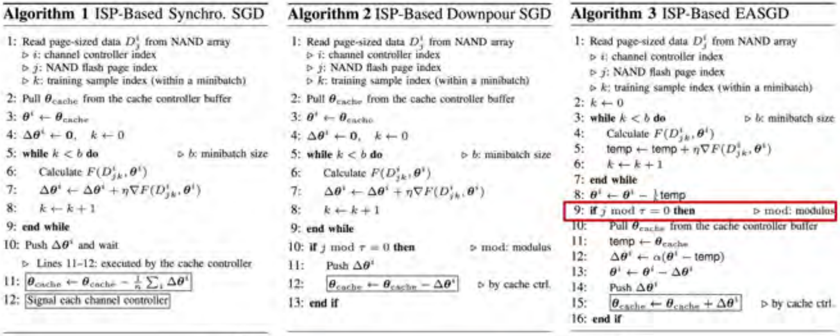
```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                        ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j$ mod $\tau = 0$ then              ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
13: end if
```

Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                        ▷ $b$: minibatch size
4:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{b}$temp
9: if $j$ mod $\tau = 0$ then               ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i - \text{temp})$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
16: end if
```



Parallel SGD Implementation on ISP-ML



Algorithm 1 ISP-Based Synchro. SGD

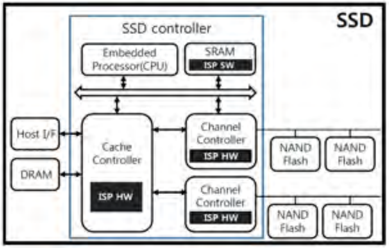
```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: Signal each channel controller
```

Algorithm 2 ISP-Based Downpour SGD

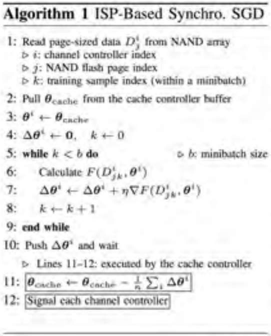
```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then                ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
13: end if
```

Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D^i_j$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                          ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D^i_{jk}, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{k}$temp
9: if $j \bmod \tau = 0$ then                 ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i -$ temp$)$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$     ▷ by cache ctrl.
16: end if
```

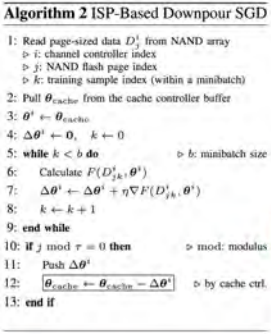


Parallel SGD Implementation on ISP-ML



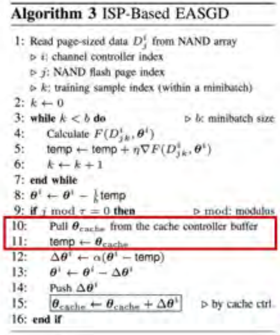
Algorithm 1 ISP-Based Synchro. SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{cache}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{cache}$
4: $\Delta\theta^i \leftarrow 0$,   $k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D^i_{jk}, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\theta^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\theta_{cache} \leftarrow \theta_{cache} - \frac{1}{n}\sum_i \Delta\theta^i}$
12: $\boxed{\text{Signal each channel controller}}$
```



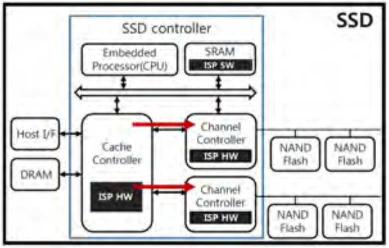
Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\theta_{cache}$ from the cache controller buffer
3: $\theta^i \leftarrow \theta_{cache}$
4: $\Delta\theta^i \leftarrow 0$,   $k \leftarrow 0$
5: while $k < b$ do                          ▷ $b$: minibatch size
6:     Calculate $F(D^i_{jk}, \theta^i)$
7:     $\Delta\theta^i \leftarrow \Delta\theta^i + \eta\nabla F(D^i_{jk}, \theta^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j$ mod $\tau = 0$ then                  ▷ mod: modulus
11:     Push $\Delta\theta^i$
12:     $\boxed{\theta_{cache} \leftarrow \theta_{cache} - \Delta\theta^i}$       ▷ by cache ctrl.
13: end if
```

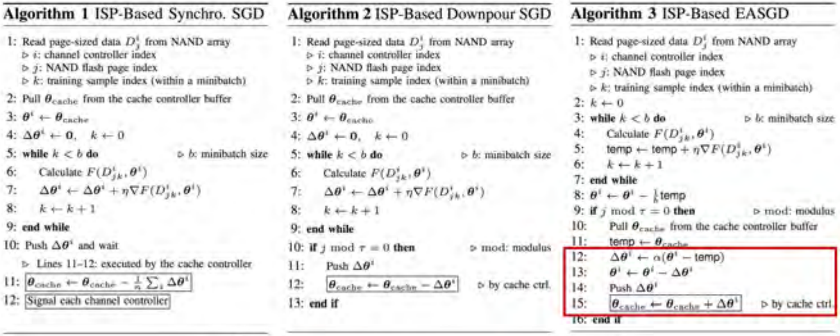


Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                          ▷ $b$: minibatch size
4:     Calculate $F(D^i_{jk}, \theta^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D^i_{jk}, \theta^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\theta^i \leftarrow \theta^i - \frac{1}{k}$temp
9: if $j$ mod $\tau = 0$ then                   ▷ mod: modulus
10:     Pull $\theta_{cache}$ from the cache controller buffer
11:     temp $\leftarrow \theta_{cache}$
12:     $\Delta\theta^i \leftarrow \alpha(\theta^i -$ temp$)$
13:     $\theta^i \leftarrow \theta^i - \Delta\theta^i$
14:     Push $\Delta\theta^i$
15:     $\boxed{\theta_{cache} \leftarrow \theta_{cache} + \Delta\theta^i}$       ▷ by cache ctrl.
16: end if
```



Parallel SGD Implementation on ISP-ML



Algorithm 1 ISP-Based Synchro. SGD

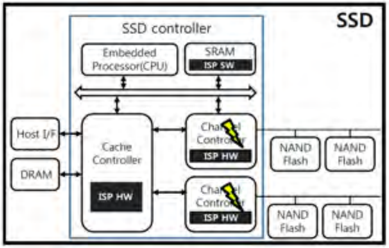
```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: Push $\Delta\boldsymbol{\theta}^i$ and wait
     ▷ Lines 11–12: executed by the cache controller
11: $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \frac{1}{n}\sum_i \Delta\boldsymbol{\theta}^i}$
12: $\boxed{\text{Signal each channel controller}}$
```

Algorithm 2 ISP-Based Downpour SGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
3: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_{\text{cache}}$
4: $\Delta\boldsymbol{\theta}^i \leftarrow \mathbf{0}, \quad k \leftarrow 0$
5: while $k < b$ do                         ▷ $b$: minibatch size
6:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
7:     $\Delta\boldsymbol{\theta}^i \leftarrow \Delta\boldsymbol{\theta}^i + \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
8:     $k \leftarrow k + 1$
9: end while
10: if $j \bmod \tau = 0$ then                ▷ mod: modulus
11:     Push $\Delta\boldsymbol{\theta}^i$
12:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} - \Delta\boldsymbol{\theta}^i}$      ▷ by cache ctrl.
13: end if
```

Algorithm 3 ISP-Based EASGD

```
1: Read page-sized data $D_j^i$ from NAND array
     ▷ $i$: channel controller index
     ▷ $j$: NAND flash page index
     ▷ $k$: training sample index (within a minibatch)
2: $k \leftarrow 0$
3: while $k < b$ do                         ▷ $b$: minibatch size
4:     Calculate $F(D_{jk}^i, \boldsymbol{\theta}^i)$
5:     temp $\leftarrow$ temp $+ \eta\nabla F(D_{jk}^i, \boldsymbol{\theta}^i)$
6:     $k \leftarrow k + 1$
7: end while
8: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \frac{1}{k}$temp
9: if $j \bmod \tau = 0$ then                 ▷ mod: modulus
10:     Pull $\boldsymbol{\theta}_{\text{cache}}$ from the cache controller buffer
11:     temp $\leftarrow \boldsymbol{\theta}_{\text{cache}}$
12:     $\Delta\boldsymbol{\theta}^i \leftarrow \alpha(\boldsymbol{\theta}^i - \text{temp})$
13:     $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i - \Delta\boldsymbol{\theta}^i$
14:     Push $\Delta\boldsymbol{\theta}^i$
15:     $\boxed{\boldsymbol{\theta}_{\text{cache}} \leftarrow \boldsymbol{\theta}_{\text{cache}} + \Delta\boldsymbol{\theta}^i}$      ▷ by cache ctrl.
16: end if
```



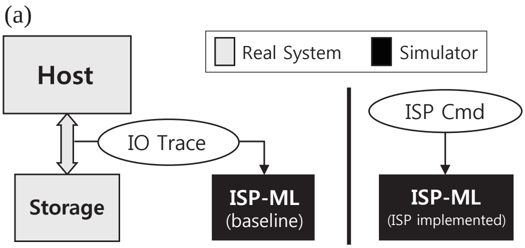
Methodology for IHP-ISP Performance Comparison

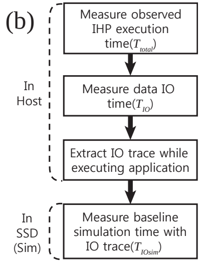
- Ideal Ways to Fairly Compare ISP and IHP
  1. Implementing ISP-ML in a real semiconductor chip
     - High chip manufacturing costs
  2. Simulating IHP in the ISP-ML framework.
     - High simulation time to simulate IHP
  3. Implementing both ISP and IHP using FPGAs.
     - Require another significant 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





(b)
In Host
Measure observed IHP execution time(T_{total})
Measure data IO time(T_{IO})
Extract IO trace while executing application
In SSD (Sim)
Measure baseline simulation time with IO trace(T_{IOsim})

- 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

Outline

Setup and Implementation

- Host specifications

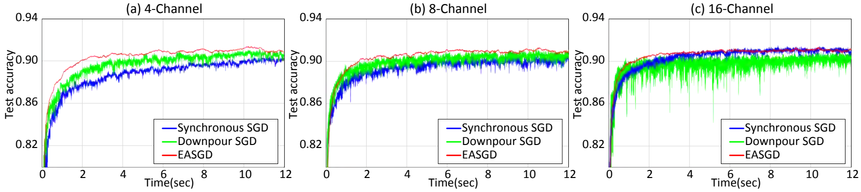
CPU	8-core Intel(R) Core i7-3770K (3.50GHz)
Main memory	DDR3 32GB RAM
Storage	Samsung SSD 840 Pro
OS	Ubuntu 14.04 LTS

- ISP-ML specifications

Embedded processor	ARM 926EJ-S (400MHz)
FTL	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)

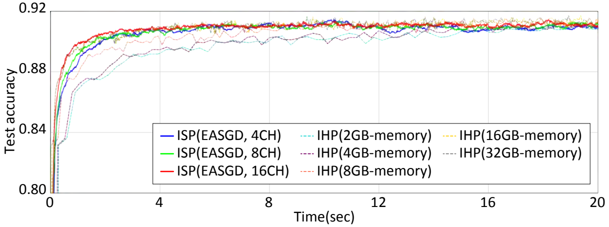
Performance Comparison:ISP-Based Optimization

- 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



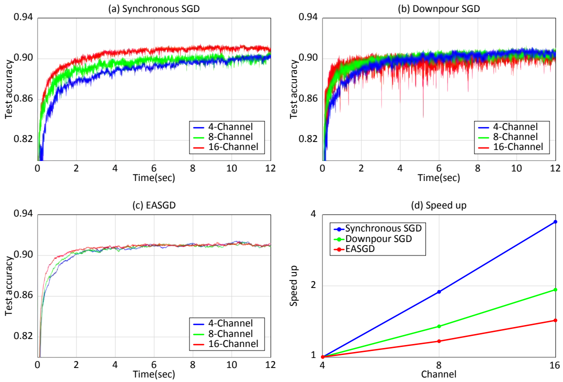
Performance Comparison:IHP versus ISP

- 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.



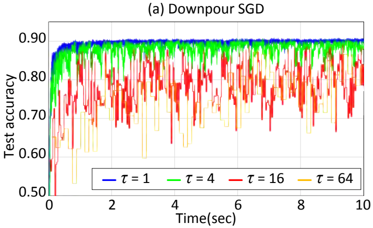
Channel Parallelism

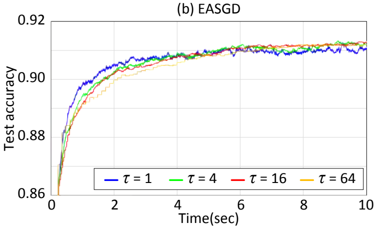
- The speed-up tends to be proportional to the number of channels.
- Because the communication overhead in ISP is negligible.
  - In distributed computing systems, communication bottleneck commonly occurs.



Effects of Communication Period in Async. SGD

- Downpour SGD
  - High speed for a low communication period [τ=1; 4]
  - Unstable for a high communication period [τ=16; 64]
- EASGD
  - Communication period ⇑, convergence speed ⇓
  - In contrast to the distributed computing system
  - Because of the low communication overhead





Experimental Results Summary

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

Outline

Parallelism in ISP

- 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
    - Distributed systems + ISP-based SSDs

Opportunities for Future Research

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

Conclusion

- 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

National Research Foundation of Korea

BrainKorea21 PLUS

Q/A