

# GlobalDataLoader in Multi DeepLearning Task

Xie Jian

I2EC, ICS, NJU

March 13, 2021

# Table of Contents

- ① Introduction
- ② Sampling Alogrithm
- ③ Global DataLoader
- ④ Experiment

# Table of Contents

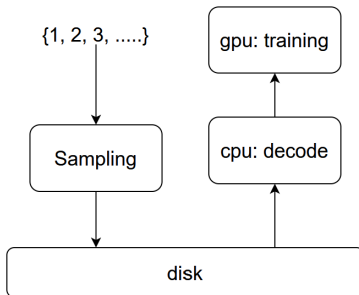
- 1 Introduction
- 2 Sampling Alogrithm
- 3 Global DataLoader
- 4 Experiment

# DataLoader in Pytorch

## Data loading

Assume there is a task training in a dataset with index set

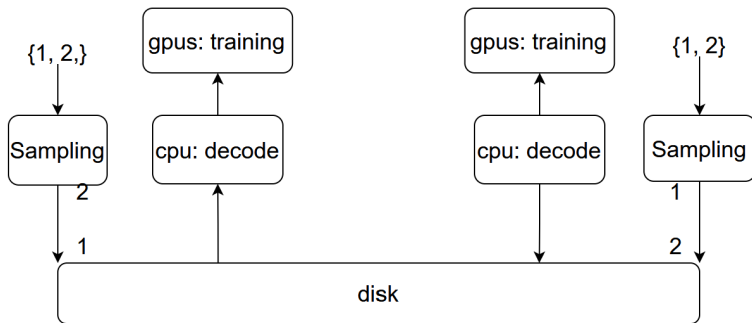
- Sampler sample some index randomly
- Worker read the data from disk
- CPU decode them
- GPU load them and training



# Multi task loading data

## Problem

The data will be repeatedly read and processed by different tasks.  
Dose it matter?



# Experiment

## Configuration

Multiple tasks with worker = 4, batch size = 32  
are trained at the same time with in a Tesla T4 with 16G memory.  
The server has 512g memory and 48 Intel(R) Xeon(R) Gold 5118 CPU.

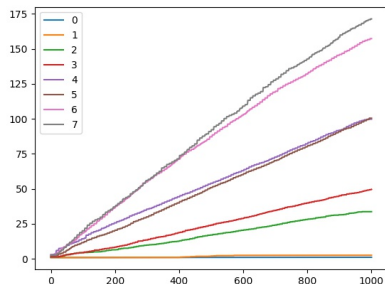


Figure: data loading time

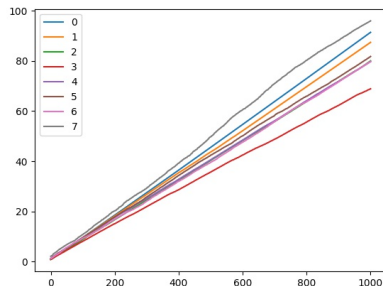


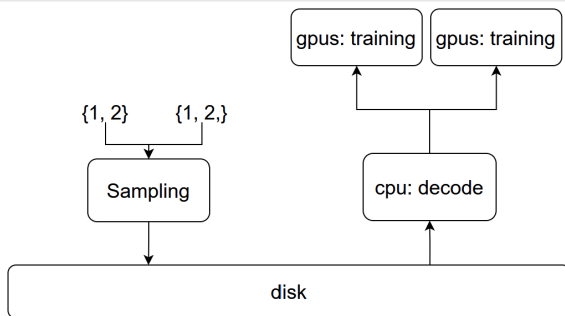
Figure: data training time

# Idea

## idea

Avoid repeatedly reading and processing:

- After shuffle, the data should be read in a similar order
- And there are a global DataLoader to read the same data only once



# Table of Contents

- ① Introduction
- ② Sampling Alogrithm
- ③ Global DataLoader
- ④ Experiment



# Sampling: problem description

## Defination

Assume there are two sets  $\{S_1, S_2\}$ , we should randomly select 2 elements  $\{e_1, e_2\}$  from them. The algorithm should to make sure to:

$$\begin{aligned} p(e_1) &= \frac{1}{|S_1|} \\ p(e_2) &= \frac{1}{|S_2|} \\ \text{Maximize}(p(e_1 = e_2)) \end{aligned} \tag{1}$$

# Independently Sampling Algorithm

## Assumption

There are two sets:  $S_1, S_2$ , and their length is  $n_1, n_2$

The intersection set of them is  $S_i$ , whose length is  $n_i$

We divide the set  $S_1$  into  $S_i$  and  $S_{d1} = S_1 - S_i$

We divide the set  $S_2$  into  $S_i$  and  $S_{d2} = S_2 - S_i$

## Example

$$S_1 = \{1, 2, 3, 4, 5\} = \{1, 2, 3\} \cup \{4, 5\}$$

$$S_2 = \{1, 2, 3, 6, 7\} = \{1, 2, 3\} \cup \{6, 7\}$$

# Independently Sampling Algorithm

## Algorithm

- $S_1$ :
  - $step_{11}$ : randomly select a set from  $S_i$  and  $S_{d1}$
  - $step_{12}$ : if the set is  $S_{d1}$ , randomly select a element  $e_1$  from  $S_{d1}$
  - $step_{13}$ : if the set is  $S_i$ , randomly select a element  $e_1$  from  $S_i$
- $S_2$ :
  - $step_{21}$ : randomly select a set from  $S_i$  and  $S_{d2}$
  - $step_{22}$ : if the set is  $S_{d2}$ , randomly select a element  $e_2$  from  $S_{d2}$
  - $step_{23}$ : if the set is  $S_i$ , randomly select a element  $e_2$  from  $S_i$

## Probability

$$\begin{aligned}
 p(e_1) &= \frac{1}{n_1}, p(e_2) = \frac{1}{n_2} \\
 p(e_1 = e_2) &= \frac{n_i}{n_1 * n_2}
 \end{aligned}
 \tag{2}$$

# Dependently Sampling Algorithm I

## Insight

The  $step_{13}$  is same as  $step_{23}$  (randomly select an element  $e$  from  $S_i$ ).

## Example

$$S_1 = \{1, 2, 3\} \cup \{4, 5\}; S_2 = \{1, 2, 3\} \cup \{6, 7\}$$

- As for  $S_1$ , select set  $\{1, 2, 3\}$  with probability  $p_1(S_i) = 0.6$
- As for  $S_2$ , select set  $\{1, 2, 3\}$  with probability  $p_2(S_i) = 0.6$
- Finally, randomly sampling element in  $\{1, 2, 3\}$
- $p(e_1 = e_2) = p_1(S_i) * p_2(S_i) = 0.36$

# Dependently Sampling Algorithm I

## Algorithm

- $S_1$ :
  - $step_{11}$ : randomly select a set from  $S_i$  and  $S_{d1}$
  - $step_{12}$ : if the set is  $S_{d1}$ , randomly select a element  $e_1$  from  $S_{d1}$
  - $step_{13}$ : if the set is  $S_i$ , go to  $step_3$
- $S_2$ :
  - $step_{21}$ : randomly select a set from  $S_i$  and  $S_{d2}$
  - $step_{22}$ : if the set is  $S_{d2}$ , randomly select a element  $e_2$  from  $S_{d2}$
  - $step_{23}$ : if the set is  $S_i$ , go to  $step_3$
- $step_3$ : randomly select a element  $e$  from  $S_i$

## Probability

$$p(e_1 = e_2) = \frac{n_i}{n_1} * \frac{n_i}{n_2} = \frac{n_i^2}{n_1 * n_2} \quad (3)$$

# Dependently Sampling Algorithm I

## Insight

The  $step_{11}$  and  $step_{21}$  are similar. We can merge them.

## Example

$$S_1 = \{1, 2, 3\} \cup \{4, 5\}; S_2 = \{1, 2, 3\} \cup \{6, 7\}$$

- Firstly, we randomly select a set from  $\{1, 2, 3\}$  and  $\{4, 5\}$
- If we choose  $\{1, 2, 3\}$ , the  $S_1$  and  $S_2$  will both select an element from  $\{1, 2, 3\}$
- If we choose  $\{4, 5\}$ , then  $S_1$  will sampling in  $\{4, 5\}$  and  $S_2$  in  $\{6, 7\}$
- $p(e_1 = e_2) = p_1(S_i) = 0.6$

# Problem

## Algorithm

- $step_1$ : randomly select a set from  $S_i$  and  $S_{d1}$
- $S_1$ :
  - $step_{12}$ : if the set is  $S_{d1}$ , randomly select a element  $e_1$  from  $S_{d1}$
  - $step_{13}$ : if the set is  $S_i$ , go to  $step_3$
- $S_2$ :
  - $step_{22}$ : if the set is not  $S_{d1}$ , randomly select a element  $e_2$  from  $S_{d2}$
  - $step_{23}$ : if the set is  $S_i$ , go to  $step_3$
- $step_3$ : randomly select a element  $e$  from  $S_i$

# Problem: $n_1 \neq n_2$

$$n_1 > n_2$$

$$S_1 = \{1, 2, 3\} \cup \{4\}; S_2 = \{1, 2, 3\} \cup \{6, 7\}$$

- $S_1$  select  $\{1, 2, 3\}$  with probability  $p_1(S_i) = 0.75$
- Then  $S_2$  select an element  $e$  from  $S_i$  with probability  $p_2(e) = p_1(S_i) * \frac{1}{3} = 0.25 > 0.2$

$$n_1 < n_2$$

$$S_1 = \{1, 2, 3\} \cup \{4, 6\}; S_2 = \{1, 2, 3\} \cup \{7\}$$

- $S_1$  select  $\{4, 6\}$  with probability  $p_1(S_{d1}) = 0.4$
- Then  $S_2$  select an element  $e$  from  $S_{d2}$  with probability  $p_2(e) = p_1(S_{d1}) * \frac{1}{1} = 0.4 > 0.25$



Case1:  $n_1 < n_2$ 

## Problem

$$(p_1(S_i) = \frac{n_i}{n_1}) > (p_2(S_i) = \frac{n_i}{n_2}) \quad (4)$$

## Approach

So in  $step_1$ , when select  $S_i$ , it should be changed  $S_{d2}$  in probability of  $p$ .

$$\begin{cases} p_2(S_i) = p_1(S_i) * (1 - p) = \frac{n_i}{n_2} \\ p_2(S_{d2}) = p_2(S_{d1}) + p_1(S_i) * p = 1 - \frac{n_i}{n_2} \end{cases} \quad (5)$$

$$p = 1 - \frac{n_1}{n_2} \quad (6)$$

# Dependently Sampling Algorithm II

## Algorithm

- *step*<sub>1</sub>: randomly select a set  $S$  from  $S_i$  and  $S_{d1}$
- $S_1$ :
  - *step*<sub>12</sub>: if the  $S$  is  $S_{d1}$ , randomly select a element  $e_1$  from  $S_{d1}$
  - *step*<sub>13</sub>: if the  $S$  is  $S_i$ , go to *step*<sub>3</sub>
- $S_2$ :
  - *step*<sub>22</sub>: if the  $S$  is  $S_{d2}$ , randomly select a element  $e_2$  from  $S_{d2}$
  - *step*<sub>23</sub>: if the  $S$  is  $S_i$ 
    - randomly let  $S = S_{d1}$  in probability of  $1 - \frac{n_1}{n_2}$
    - if  $S$  is  $S_{d2}$ , randomly select a element  $e_2$  from  $S_{d2}$
    - if  $S$  is  $S_i$ , go to *step*<sub>3</sub>
- *step*<sub>3</sub>: randomly select a element  $e$  from  $S_i$

## Probability

$$p(e_1 = e_2) = p_1(S_i) * \left(\frac{n_1}{n_2}\right) = \frac{n_i}{n_2}$$

Case2:  $n_1 > n_2$ 

## Problem

$$(p_1(S_{d1}) = 1 - \frac{n_i}{n_1}) > (p_2(S_{d2}) = 1 - \frac{n_i}{n_2}) \quad (7)$$

## Approach

So in  $step_1$ , when select  $S_{d1}$ , it should be changed  $S_i$  in probability of  $p$ . The equation is

$$\begin{cases} p_2(S_i) = p_1(S_i) + p_2(S_{d1}) * p = \frac{n_i}{n_2} \\ p_2(S_{d2}) = p_2(S_{d1}) * (1 - p) = 1 - \frac{n_i}{n_2} \end{cases} \quad (8)$$

$$p = 1 - \frac{n_1 * (n_2 - n_i)}{n_2 * (n_1 - n_i)} \quad (9)$$

# Dependently Sampling Algorithm II

## Algorithm

- $step_1$ : randomly select a set  $S$  from  $S_i$  and  $S_{d1}$
- $S_1$ :
  - $step_{12}$ : if the  $S$  is  $S_{d1}$ , randomly select a element  $e_1$  from  $S_{d1}$
  - $step_{13}$ : if the  $S$  is  $S_i$ , go to  $step_3$
- $S_2$ :
  - $step_{22}$ : if the  $S$  is  $S_{d2}$ , randomly select a element  $e_2$  from  $S_{d2}$ 
    - randomly let  $S = S_i$  in probability of  $1 - \frac{n1*(n2-ni)}{n2*(n1-ni)}$
    - if  $S$  is  $S_{d2}$ , randomly select a element  $e_2$  from  $S_{d2}$
    - if  $S$  is  $S_i$ , randomly select a element  $e_2$  from  $S_i$
  - $step_{23}$ : if the  $S$  is  $S_i$ , go to  $step_3$
- $step_3$ : randomly select a element  $e$  from  $S_i$

## Probability

$$p(e_1 = e_2) = p_1(S_i) = \frac{n_i}{n_1}$$

# Why not shuffle $S_1 \cup S_2$

## Probability

$$p(e_1 = e_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} < \frac{|S_1 \cap S_2|}{\max(|S_1|, |S_2|)}$$

randomly select in  $S_1 \cup S_2$ : "starvation"

if  $|S_1| = 99, |S_2| = 1$ , then  $p(e \in S_1) = 0.99, p(e \in S_2) = 0.01$   
So task2 may be starving

shuffle  $S_1 \cup S_2$ : offset

$$S_1 = \{2, 3\}, S_2 = \{1, 2, 3\}$$

$$\{2, 3\}$$

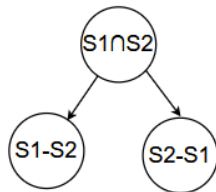
$$\{1, 2, 3\}$$

$$\{1, 2, 3\}$$

# Sampling Tree: two sets

## Algorithm

- case1: Both randomly sample in  $S_1 \cap S_2$ 
  - $p_1 < \frac{n_i}{n_1}$
  - $p_2 < \frac{n_1}{n_2}$
- case2: For  $S_1$ , sample in  $S_1 \cap S_2$ . For  $S_2$ , sample in  $S_2 - S_1$ 
  - $p_1 < \frac{n_i}{n_1}$
  - $p_2 > \frac{n_1}{n_2}$
- case3: For  $S_1$ , sample in  $S_1 - S_2$ . For  $S_2$ , sample in  $S_2 - S_1$ 
  - $p_1 > \frac{n_i}{n_1}$

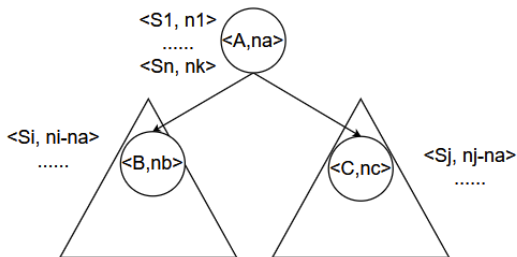


# Sampling Tree: n sets

## Algorithm: parent node

There are two sets:  $S_{child}, S_{parent}$ .

- $p_1 < \frac{n_a}{n_1}, S_{parent} = S_{parent} \cup S_1$   
.....
- $p_j > \frac{n_{j-1}}{n_j}, S_{child} = S_{child} \cup S_j$
- $p_{j+1} < \frac{n_{j-1}}{n_{j+1}}, S_{parent} = S_{parent} \cup S_{j+1}$



# Prove

Basis: For two sets  $\{S_1, S_2\}$ :

$$\begin{aligned} p(e_1) &= \frac{1}{|S_1|}, p(e_1) = \frac{1}{|S_2|} \\ p(e_1 = e_2) &= \frac{1}{\max(|S_1|, |S_2|)} \end{aligned} \quad (10)$$

Induction:

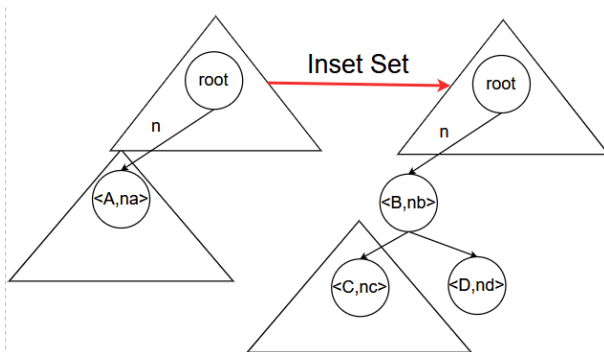
If  $\{S_1, S_2, \dots, S_k\}$  satisfies:

$$\begin{aligned} \forall i, p(e_i) &= \frac{1}{|S_i|} \\ \forall i, j, p(e_i = e_j) &= \frac{1}{\max(|S_i|, |S_j|)} \end{aligned} \quad (11)$$

then  $\{S_1, S_2, \dots, S_k, S_{k+1}\}$  sets is also satisfies



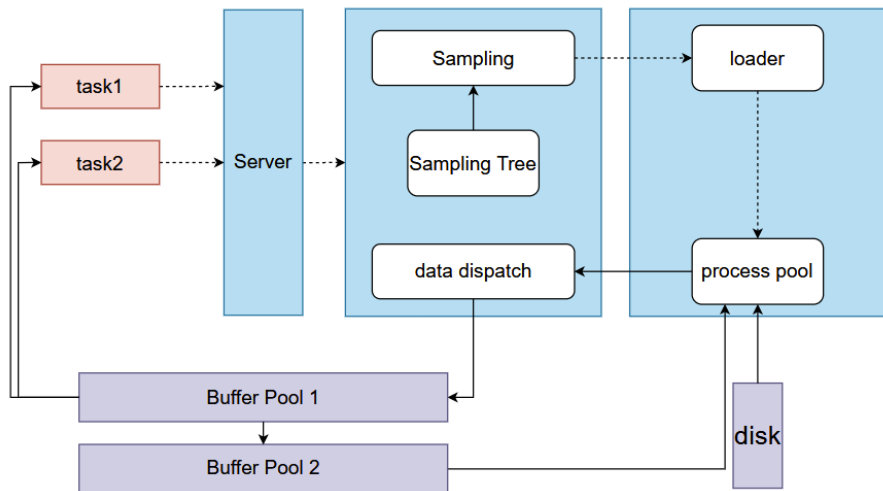
# Induction



# Table of Contents

- ① Introduction
- ② Sampling Alogrithm
- ③ Global DataLoader
- ④ Experiment

# Architecture



# Server and Loader

## Server

- receive task <task name, index set> and return head address
- heartbeat

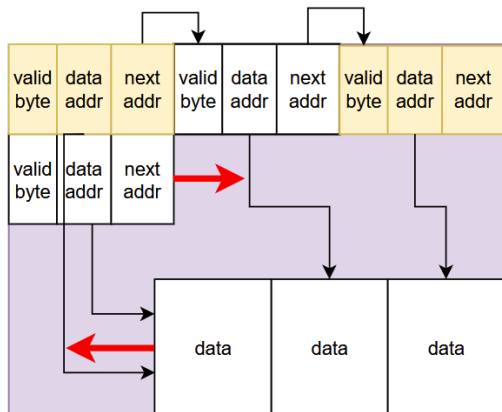
## Loader

- check Buffer Pool2
- if miss, read from disk

# Buffer Pool: Data Structure

## data

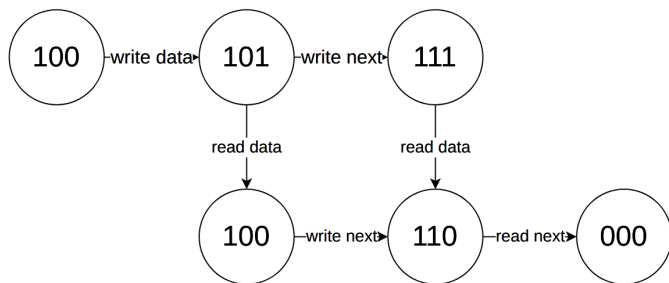
- There are two kinds of nodes: inode and datanode, and they are fixed size.
- Every task has a head inode address



# Buffer Pool: Valid Byte

## valid byte

- used bit: If the used bit is equal to 1, this inode is used
- next bit: If the next bit is equal to 1, the next addr is valid
- data bit: If the data bit is equal to 1, the data addr is valid



# Buffer Pool: Evict

## inode

if used bit is equal to 0, then it can be evicted

## data node

if all reference nodes of datanode is unused, then it can be evicted to Buffer Pool2

## Buffer Pool 2

LRU

# Table of Contents

- 1 Introduction
- 2 Sampling Alogrithm
- 3 Global DataLoader
- 4 Experiment



# Experiment

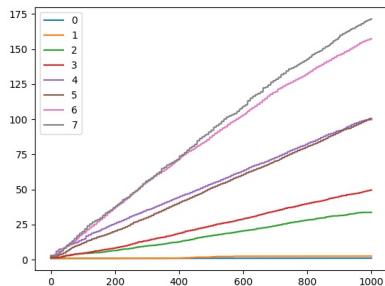


Figure: time

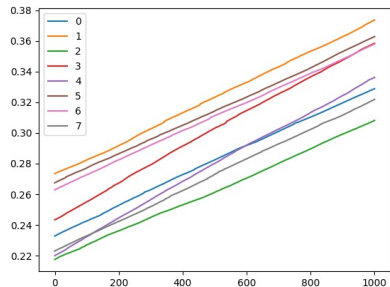


Figure: time with GlobalDataLoader