GlobalDataLoader in Multi DeepLearning Task

Xie Jian

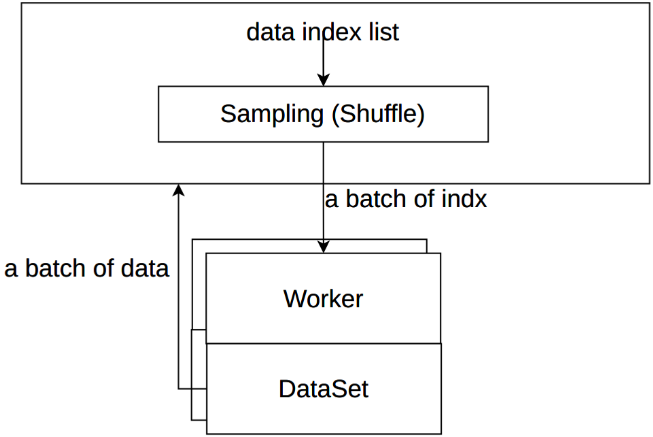
I2EC, ICS, NJU

March 8, 2021

Table of Contents

Table of Contents

DataLoader in Pytorch



Problem: Repeated Reading and Processing

Situation

To compare the performance of different algorithms, Many DeepLearning tasks are training in the same Dataset.

Problem

Every task has its own DataLoader. So the data will be repeatedly read and processed by different tasks.

Result

As the number of tasks increases, so does the training time. And what increases is the time to load the data

Experiment

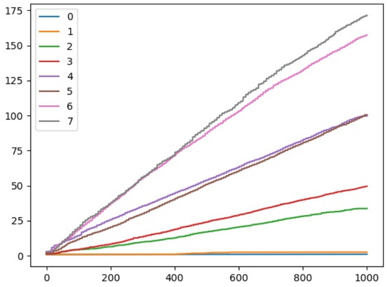


Figure: data loading time

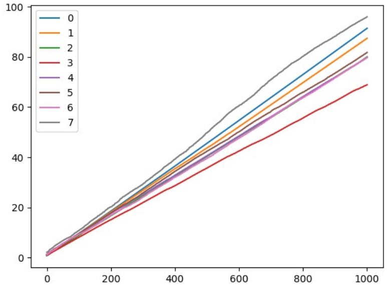
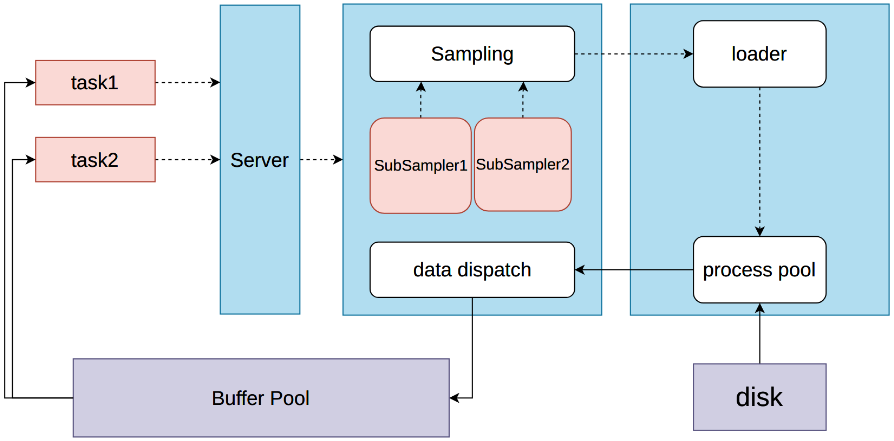


Figure: data training time

Table of Contents

Architecture



Sampling: problem description

Defination

For a single task, the sampler needs to select an index from the index set S.
Similarly, for multiple tasks, the sampler needs to select an index from multiple sets $\{S_1, S_2, ...\}$

Requirments

- The index in the set S should be randomly sampled. The probability of the index being selected is $1/|S|$
- Duplicate indexes need to be merged
- There can be no problem of "starvation"

Assumption 1

Requirments

- There are only two sets $\{S_1, S_2\}$
- S_1 is same as S_2. And the length of S_1 and S_2 is n

Solution 1

description

In order to avoid the problem of "starvation", we sampling the idx through polling.

steps

- First, We randomly select an idx i_1 from the S_1.
- Because S_1 and S_2 are the same, we don't need to sample S_2

Assumption 2

Requirments

- There are only two sets $\{S_1, S_2\}$
- S_1 and S_2 are equal in length, which is n
- The intersection of S_1 and S_2 is S_i, whose length is n_i

Solution 2

steps

- First, We randomly select an idx i_1 from the S_1.
- If $i_1 \in S_i$ then $i_2 = i_1$
- If $i_1 \notin S_i$ then we randomly select an idx i_2 from the $S_2 - S_1$

Proving

S1

$p(i_1) = \frac{1}{n}$

S2

If $i_2 \in S_i$,

$$p(i_2) = \frac{n_i}{n} * \frac{1}{n_i} = \frac{1}{n}$$

If $i_2 \notin S_i$,

$$p(i_2) = \frac{n - n_i}{n} * \frac{1}{n - n_i} = \frac{1}{n}$$

Assumption 3

Requirments

- There are only two sets $\{S_1, S_2\}$
- S_1 is different from S_2, and their length are n_1 and n_2
- The intersection of S_1 and S_2 is S_i, whose length is n_i

Solution 3

If we use Solution2.

S2

If $i_2 \in S_i$,

$$p(i_2) = \frac{n_i}{n_1} * \frac{1}{n_i} = \frac{1}{n_1}$$

If $i_2 \notin S_i$,

$$p(i_2) = \frac{n_1 - n_i}{n_1} * \frac{1}{n2 - n_i} = \frac{n_1 - n_i}{n_1 * (n_2 - n_i)}$$

Solution 3

if $n_1 < n_2$, then $p(i_2 \in S_i) > \frac{1}{n_2}$. So in step 2 of Solution2, we should randomly select a idx in $S_2 - S_i$ in probability of x.

I

f $i_2 \in S_i$,

$$p(i_2) = \frac{n_i}{n_1} * (1 - x) * \frac{1}{n_i} = \frac{1}{n_2}$$

If $i_2 \notin S_i$,

$$p(i_2) = \frac{n_1 - n_i}{n_1} * \frac{1}{n2 - n_i} + \frac{n_i}{n_1} * x * \frac{1}{n2 - n_i} = \frac{1}{n_2}$$

then

$$x = \frac{n_2 - n_1}{n_2}$$

Solution 3

if $n_1 > n_2$, then $p(i_2 \notin S_i) > \frac{1}{n_2}$. So in step 3 of Solution2, we should randomly select a idx in S_i in probability of x.

I

f $i_2 \in S_i$,

$$p(i_2) = \frac{n_i}{n_1} * \frac{1}{n_i} + x * \frac{n1 - n_c}{n1} * \frac{1}{n_i} = \frac{1}{n_2}$$

If $i_2 \notin S_i$,

$$\frac{n_1 - n_i}{n_1} * \frac{1}{(n_2 - n_i)} * (1 - x) = \frac{1}{n_2}$$

then

$$x = 1 - \frac{n_1 * (n_2 - n_i)}{n_2 * (n_1 - n_i)}$$

Solution 3

steps

- First, We randomly select an idx i_1 from the S_1.
- If $i_1 \in S_2$ and $n_1 < n_2$, randomly sample in $S_2 - S_i$ with a probability of $\frac{n_2 - n_1}{n_2}$
- If $i_1 \in S_2$ and $n_1 > n_2$, do nothing
- If $i_1 \notin S_2$ and $n_1 > n_2$ randomly sample in S_i with a probability of $1 - \frac{n_1 * (n_2 - n_i)}{n_2 * (n_1 - n_i)}$
- If $i_1 \notin S_2$ and $n_1 < n_2$ do nothing

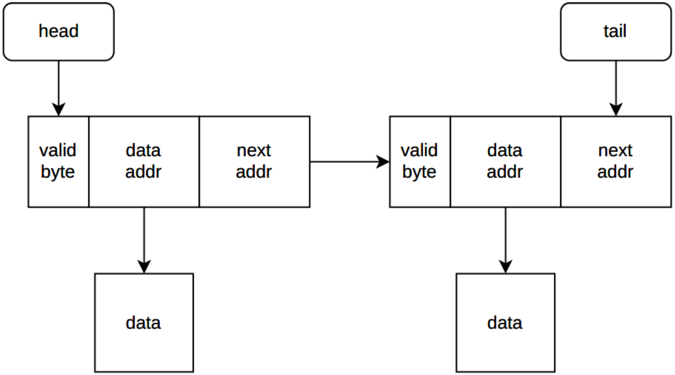
Sampling Tree

Insert

Delete

Sampling

Buffer Pool: Data Structure

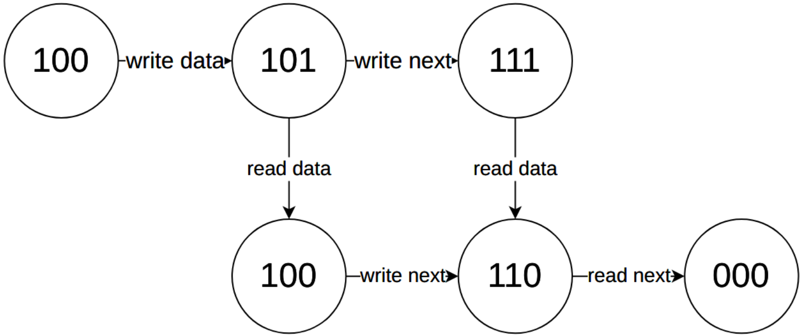


Buffer Pool: Valid Byte

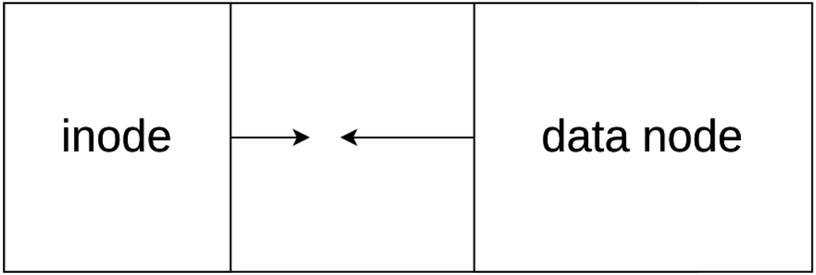
valid byte

- data bit: If the data bit is equal to 1, the data addr is valid. Otherwise invalid
- next bit: If the next bit is equal to 1, the next addr is valid. Otherwise invalid
- used bit: If the used bit is equal to 1, this inode is used by some tasks

Buffer Pool: Automata



Buffer Pool: Address Space



Buffer Pool: allocate inode

case1

There is enough free space to allocate

```
    if inode_tail + inode_size > data_head:
        return inode_tail
```

case2

Free Some unused inode

```
    for head in all_heads:
        if check_free(head) is True:
            return head
```

Buffer Pool: allocate data node

case1

There is enough free space to allocate

```
if inode_tail + inode_size > data_head:
    return inode_tail
```

case2

Free Some unused datanode

```
free = True
for datanode in all_datanodes:
    for ref in refs of datanode:
        if databit(ref) == 0 && dataaddr(ref) == datanode:
            free = False
            break
    if free is True:
        return datanode
```

Table of Contents

Experiment

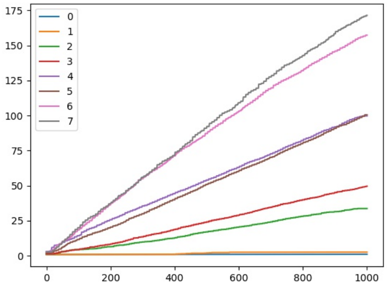


Figure: time

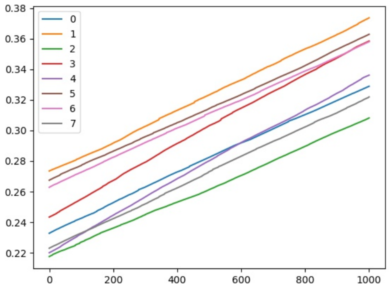


Figure: time with GlobalDataLoader