GlobalDataLoader in Multi DeepLearning Task

Xie Jian

I2EC, ICS, NJU

April 7, 2021

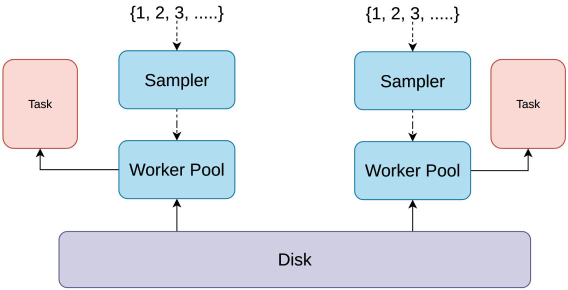
Table of Contents

Table of Contents

Introduction

Data loading

- Sampler sample some index randomly
- Worker read the data from disk and decode them
- Task fecth data and start training



Problem

Problem

The data will be repeatedly read and processed by different tasks.

Configuration

Multiple tasks with worker = 4, batch size = 32
GPU: Tesla T4 with 16G memory, CPU: 48 Intel(R) Xeon(R) Gold 5118

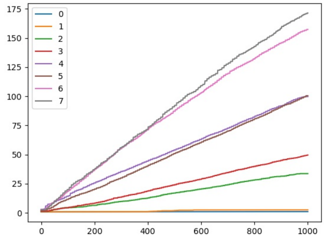


Figure: data loading time

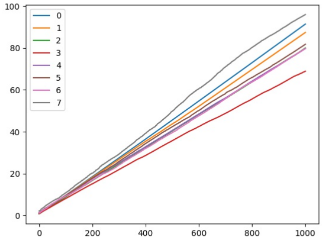
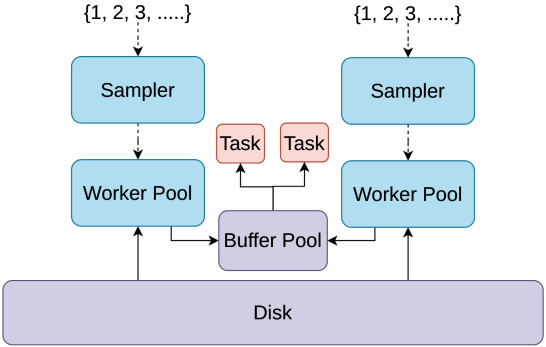


Figure: data training time

Optimization

Global Buffer Pool

- If data in buffer pool, there is no need to read data from disk



Problem

- Random replacement algorithm

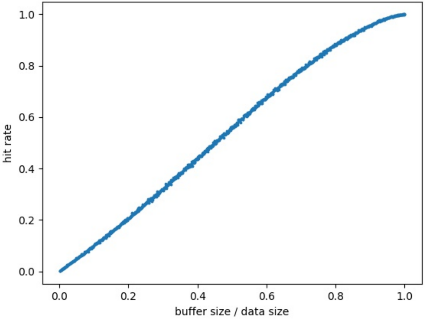


Figure: hit rate with buffer size

Optimization

Global Sampler

- Make the sampled elements have a greater probability of being equal

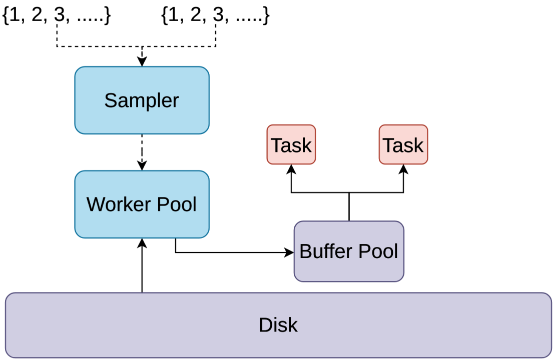


Table of Contents

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:

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

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 $S_{d2} = S_2 - S_i$
Sample(S): randomly select an element in S

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

S1

- *step*$_{11}$:randomly select a set from S_i and S_{d1}
- *step*$_{21}$:if S_{d1}, e_1=Sample(S_{d1})
- *step*$_{31}$:if S_i, e_1=Sample(S_i)

S2

- *step*$_{12}$:randomly select a set from S_i and S_{d2}
- *step*$_{22}$:if S_{d2}, e_2=Sample(S_{d2})
- *step*$_{32}$:if S_i, e_2=Sample(S_i)

Probability

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

Dependently Sampling Algorithm I

Insight

The $step_{31}$ is same as $step_{32}$ (randomly select an element e from S_i).

Example

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

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

Dependently Sampling Algorithm I

Algorithm

- $step_0$: $e_i = \text{Sample}(S_i)$

S1

- $step_{11}$:randomly select a set from S_i and S_{d1}
- $step_{21}$:if S_{d1},$e_1 = \text{Sample}(S_{d1})$
- $step_{31}$:if S_i, $e_1 = e_i$

S2

- $step_{12}$:randomly select a set from S_i and S_{d2}
- $step_{22}$:if S_{d2}, $e_2 = \text{Sample}(S_{d2})$
- $step_{32}$:if S_i, $e_2 = e_i$

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 $step_{12}$ 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 S1 and S2 will both select an element from $\{1, 2, 3\}$
- If we choose $\{4, 5\}$, then S1 will sampling in $\{4, 5\}$ and S2 in $\{6, 7\}$
- $p(e_1 = e_2) = p_1(S_i) = 0.6$

Dependently Sampling Algorithm II

Algorithm

- $step_0$: $e_i = \text{Sample}(S_i)$
- $step_1$: randomly select a set from S_i and S_{d1}

S1

- $step_{21}$: if S_{d1},e_1=Sample(S_{d1})
- $step_{31}$: if S_i, $e_1 = e_i$

S2

- $step_{22}$: if S_{d1},e_2=Sample(S_{d2})
- $step_{32}$: if S_i, $e_2 = e_i$

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: n1 < n2

Problem

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

Approach

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

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

$$p = 1 - \frac{n_1}{n_2} \tag{6}$$

Dependently Sampling Algorithm II

Algorithm

- $step_0$: $e_i = \text{Sample}(S_i)$
- $step_1$: randomly select a set from S_i and S_{d1}

S1

- $step_{21}$:if S_{d1}, e_1=Sample(S_{d1})
- $step_{31}$:if S_i, $e_1 = e_i$

S2

- $step_{22}$: if S_{d1}, e_2=Sample(S_{d2})
- $step_{32}$: if S_i:
  - let $S = S_{d1}$ in probability of $1 - \frac{n_1}{n_2}$
  - if S_{d1}, e_2=Sample(S_{d2})
  - if S_i, $e_2 = e_i$

Probability

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

Case2: n1 > n2

Problem

$$(p_1(S_{d1}) = 1 - \frac{n_i}{n_1}) > (p_2(S_{d_2}) = 1 - \frac{n_i}{n_2}) \tag{7}$$

Approach

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

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

$$p = 1 - \frac{n1 * (n2 - n_i)}{n2 * (n1 - n_i)} \tag{9}$$

Dependently Sampling Algorithm II

Algorithm

- $step_0$: $e_i = \text{Sample}(S_i)$
- $step_1$: randomly select a set from S_i and S_{d1}

S1

- $step_{21}$:if S_{d1}, e_1=Sample(S_{d1})
- $step_{31}$:if S_i, $e_1 = e_i$

S2

- $step_{22}$: if S_{d1}:
  - let $S = S_i$ in probability of $1 - \frac{n1*(n2-n_i)}{n2*(n1-n_i)}$
  - if S_{d1}, e_2=Sample(S_{d2})
  - if S_i, $e_2 = e_i$
- $step_{32}$: if S_i, $e_2 = e_i$

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: there are two sets: S_1, S_2

- $A = S_1 \cap S_2$
- $B = S_1 - A$
- $A = S_2 \cap A$

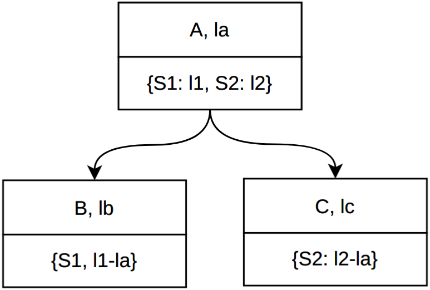


Figure: Sampling Tree

Sampling Tree: insert S_3

- $A \subset S_3$
- Insert $\{S_3 : l_3\}$ in A ascending order
- $S_3 = S_3 - A$
- Insert S_3 in subtree that has the largest intersection with S_3

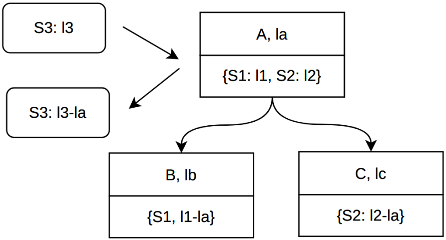


Figure: Sampling Tree

Sampling Tree: Insert S_3

- $B \not\subset S_3$
- Create new node: $D = S_3 \cap B$
- $E = B - D$
- $F = S_3 - D$

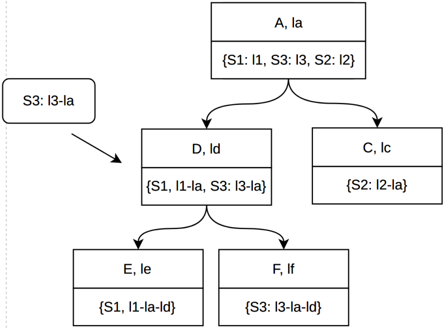


Figure: Sampling Tree

Sampling Tree: Delete S_3

- Delete S_3 in root
- Recursively let the subtree delete S_3
- Until reaching the leaf node, then delete it
- The corresponding parent node merges the child nodes

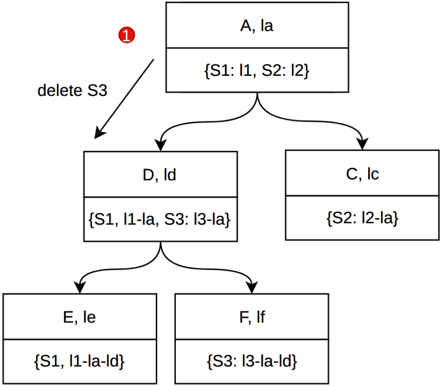


Figure: Sampling Tree

Sampling Tree: Sampling in root

- Split U into *parent* and *child*
    - $p < \frac{l_a}{l_1}$, $parent \cup \{S1\}$
    - ......
    - $p > \frac{l_i}{l_{i+1}}$, $child \cup \{S_i, S_{i+1}, \ldots\}$
- Sampling e in A, which represents the sampling result of *parent*
- Push down *child*
- Because $e \notin A$, so we need to push down e and the collection U_e containing it

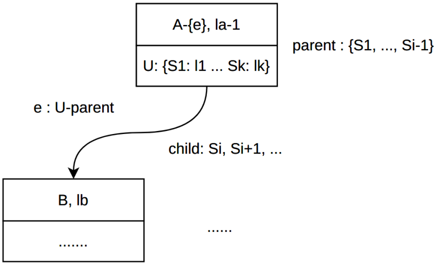


Figure: Sampling Tree

Sampling Tree: Sampling in child

- Split child into *parent* and *child*
- Sampling e in A, which represents the sampling result of *parent*
- Push down *child*
- Because $e \notin A$, so we need to push down e and the collection containing it
- if $U_e \subset U$, then add e in this node

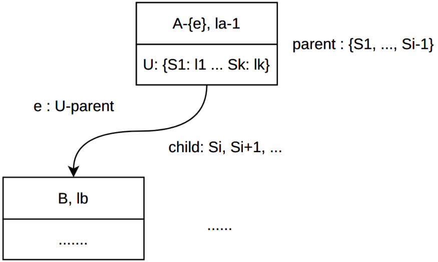
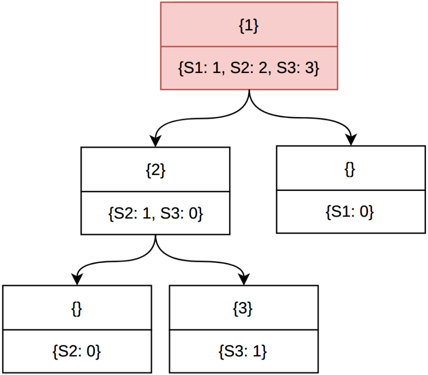
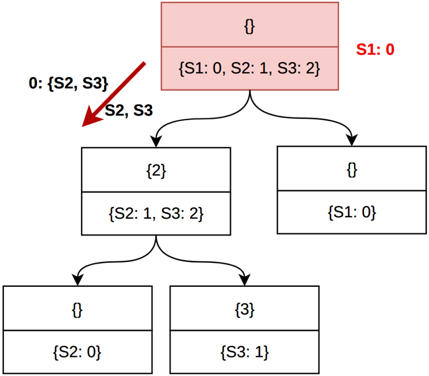


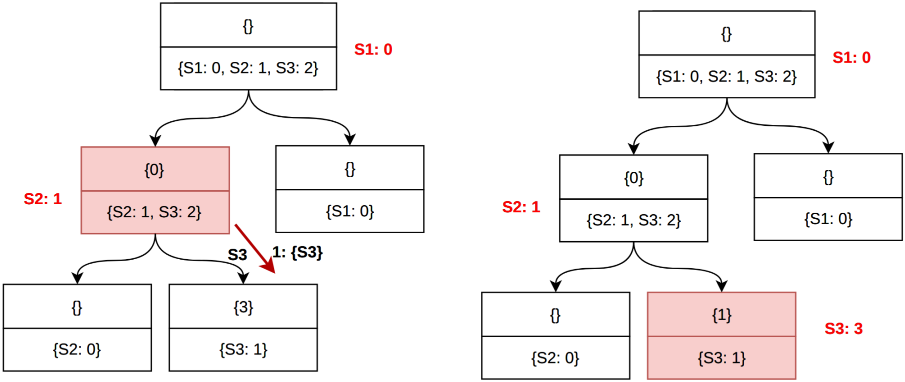
Figure: Sampling Tree

Example


{1}
{S1: 1, S2: 2, S3: 3}
{2}
{S2: 1, S3: 0}
{}
{S1: 0}
{}
{S2: 0}
{3}
{S3: 1}


{}
{S1: 0, S2: 1, S3: 2}
S1: 0
0: {S2, S3}
S2, S3
{2}
{S2: 1, S3: 2}
{}
{S1: 0}
{}
{S2: 0}
{3}
{S3: 1}

Example


{}
{S1: 0, S2: 1, S3: 2}
S1: 0
{0}
{S2: 1, S3: 2}
S2: 1
{}
{S1: 0}
S3
1: {S3}
{}
{S2: 0}
{3}
{S3: 1}
{}
{S1: 0, S2: 1, S3: 2}
S1: 0
{0}
{S2: 1, S3: 2}
S2: 1
{}
{S1: 0}
{}
{S2: 0}
{1}
{S3: 1}
S3: 3

Sampling Tree: randomly prove

Basis

1-path: $p(e) = \frac{1}{l}$

Induction

Assume for k-path, $p(e) = \frac{1}{l}$
For (k+1)-path, add a root node: $\{A : l_a\}$.
Case1: sampling in A

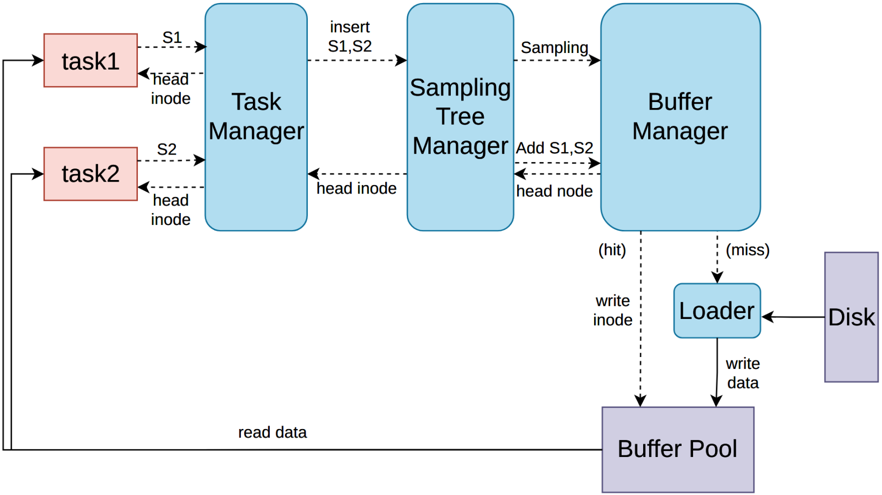
$$p(e) = p(A) * \frac{1}{l_a} = \frac{l_a}{l + l_a} * \frac{1}{l_a} = \frac{1}{l + l_a}$$

Case2: sampling in subtree

$$p(e) = (1 - p(A)) * \frac{1}{l} = \frac{l}{l + l_a} * \frac{1}{l} = \frac{1}{l + l_a}$$

Table of Contents

Architecture



Task Manager and Loader

Task Manager

- recieve task <task name, index set> and return head address
- send hearbeat

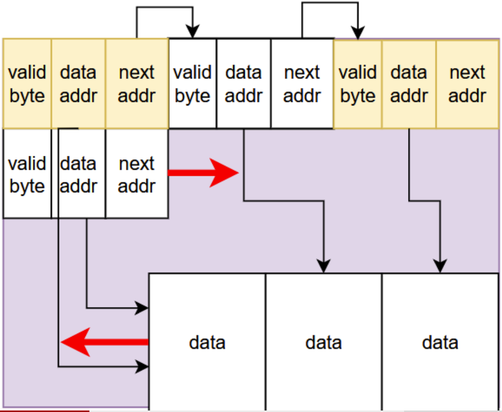
Loader

- process pool
- read data from disk and decode them
- write them in buffer pool

Buffer Pool: Data Structure

data

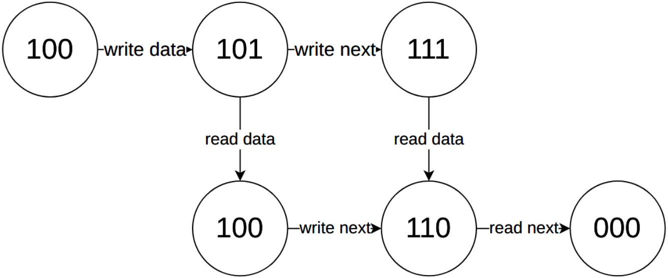
- There are two kinds of nodes: inode and datanode, and they are fixed size.
- Every task has a head inode address



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 Manager

- BM is responsible for maintaining three tables
- BM is responsible for freeing useless nodes

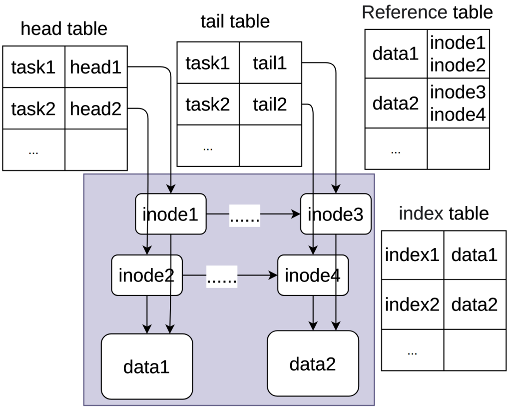


Figure: Sampling Tree

which data shound be free

Expection diff

The difference between
the number of times data has been quoted and
the number of times data should be quoted

Example

$S1 = 1, 2, 3$, $S2 = 1, 2$
The Sampling result is S1: 3, S2: 2
$ExpectionDiff(3) = 1 - 1 = 0,$
$ExpectionDiff(2) = 2 - 1 = 1$

Answer

Choose the smallest Expection diff

Table of Contents

Hit rate Experiment

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
$hitrate = \frac{hit}{n_i}$
$bufferSize = k$, which means that the buffer can have k datanode, and we ignore the size of the inode

Number of orphans

$p(e_1 = e_2 | e_2 \in S_i) = \frac{n_1}{n_2}$
Assume $p = \frac{n_1}{n_2}$

$$\begin{aligned} N &= \sum_{i=0}^{n_i} (1 - \frac{n_1 - i}{n_2 - i}) \\ &\geq \sum_{i=0}^{n_1} (1 - \frac{n_1 - i}{n_2 - i}) \\ &= \sum_{i=0}^{p*n_2} (1 - \frac{p * n_2 - i}{n_2 - i}) \\ &\approx \int_{i=0}^{p*n_2} (1 - \frac{p * n_2 - i}{n_2 - i}) \\ &= n_2 * (p - 1) ln^{1-p} \end{aligned} \tag{10}$$

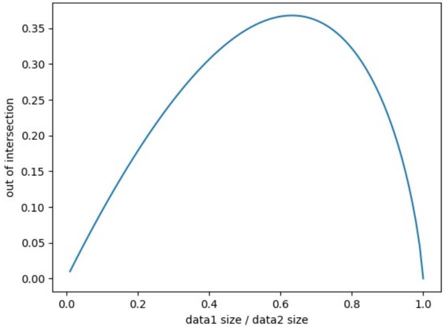


Figure:

Hit rate

$$buffer\_size = \frac{k}{n_2}$$

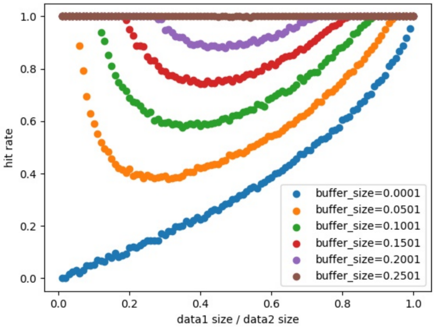


Figure:

Time Experiment

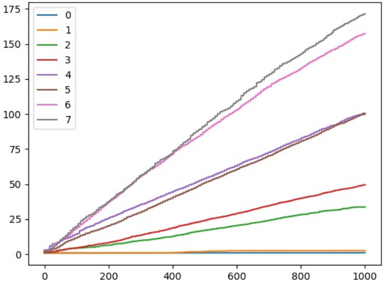


Figure: time

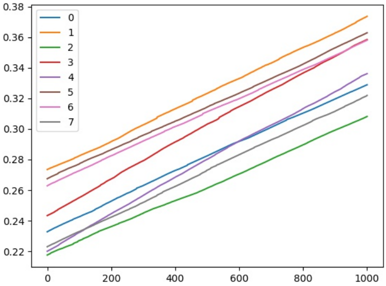


Figure: time with GlobalDataLoader

Correctness Experiment