https://www.youtube.com/live/JMUWSdSD1Uk первое видео
https://contest.yandex.ru/contest/82512/enter Контест отбора
https://new.contest.yandex.ru/contests/82514/enter Дипломная работа

# ОТБОР ‘ 371/400 , 250 points минимально для прохода

## A. Предсказание ошибок [csv файл] (100p)

Полное решение
Ограничение времени 20 секунд
Ограничение памяти 256 Мб
Предсказание ошибок
Нерадивый ML-инженер Тензорослав обучал линейную регрессию на некотором датасете. Однако, он не был знаком с градиентными методами оптимизации, поэтому Тензорослав решил искать оптимум с помощью случайного поиска. Для этого он сгенерировал несколько случайных векторов весов и считал среднеквадратичную ошибку (MSE) для каждого вектора. Однако в какой-то момент Тензорослав случайно удалил выборку, на которой он считал MSE.

Помогите Тензорославу закончить начатое и предскажите значение MSE для тех векторов весов, которые он еще не успел обработать.

О датасете
Вам предлагаются три файла:

train_weights.csv - содержит значения весов в колонках W0–W9 (10-мерные векторы) и соответствующие значения MSE в колонке MSE
test_weights.csv - содержит тестовые векторы весов (для которых нужно предсказать MSE)
example - пример корректной посылки в контест
Что нужно сделать
Необходимо предсказать колонку MSE для векторов из файла test_weights.csv и загрузить файл answers в формате, аналогичном примеру.

Критерий оценки:
Корень из средней квадратичной логарифмической ошибки (RMSLE).

Баллы рассчитываются по формуле:
100 * max(min((0.3 - RMSLE) / 0.1, 1), 0)

Для максимального балла (100) необходимо достичь RMSLE ≤ 0.2.

Требуемый формат вывода
Файл answers (без пробелов в названии) в формате JSON:


```
[
  {
    "W0": 0.465,
    "W1": 0.582,
    "W2": 1.071,
    "W3": -1.889,
    "W4": -0.92,
    "W5": -1.084,
    "W6": -1.003,
    "W7": -0.766,
    "W8": -0.858,
    "W9": 0.665,
    "MSE": 40.593
  },
  ...
]
```


**Files** - train_weights.csv , test_weights.csv , example , answer.json

**РЕШЕНИЕ** - answer.json

## B. Лаборатория биокибернетики [csv файл] (71p)

Полное решение
Ограничение времени 20 секунд
Ограничение памяти 200 Мб
Исследователи из лаборатории биокибернетики разработали алгоритм для классификации биологических объектов.

Однажды младший научный сотрудник предложил простое определение: «Объект класса X — это организм с двумя конечностями и без перьевого покрова». Но старший коллега, известный своим остроумием, принёс в лабораторию ощипанную курицу и заявил: «По вашему определению, это тоже объект X!». Пришлось уточнить критерий: «...и с плоскими когтями».

Чтобы автоматизировать классификацию, исследователи закодировали признаки организмов латинскими буквами от A до I. Они не раскрыли, что означает каждый признак, поэтому их интерпретация остаётся неизвестной.

Позже команда собрала данные:

Младший сотрудник подготовил обучающую выборку с метками.
Старший — тестовую выборку, но забыл указать классы объектов и ушёл на перерыв.
Задача: Используя обучающие данные, предскажите метки классов для тестовой выборки.

О датасете
Вам предлагаются три файла, train.csv, test.csv и example.csv:

Файл train.csv содержит признаки обучающей выборки и колонку с разметкой target. Значение 1 в этой колонке соответствует человеку, а 0 - нечеловеку.
Файл test.csv содержит признаки тестовой выборки.
Файл example.csv содержит пример корректной посылки в контест.
Таким образом, вам нужно предсказать колонку target для объектов из файла test.csv.

Что нужно сделать
От вас требуется загрузить в систему файл answers.csv в формате, аналогичном файлу example.csv с предсказаниями для объектов тестовой выборки. В качестве целевой метрики используется ROC-AUC. Баллы за это задание рассчитываются по формуле:
$100*\max(\min((AUC-0.8)/0.08,1),0)$,

где AUC - значение ROC-AUC ваших предсказаний. Таким образом, для максимального числа баллов необходимо набрать
$AUC \geq 0.88$.

Ответ

**Files** - train.csv , test.csv , example.csv , answer.json

**РЕШЕНИЕ** - answer.json

## C. Доставка заказов (100р)

Полное решение
Ограничение времени 2 секунды
Ограничение памяти 512 Мб
Ввод стандартный ввод или input.txt
Вывод стандартный вывод или output.txt

Яндекс тестирует автономного курьера-ровера и хочет понять, насколько быстро он сможет развозить заказы по городу.

Город задан прямоугольной решёткой
$n$
×
$m$
$n \times m$. Для удобства все адреса поделены на
26
26 типов и обозначены строчными латинскими буквами от ‘a’ до ‘z’.

Диспетчер выдал роверу список доставок в виде строки
$s$
$s$:
$i$
$i$-й символ — это тип адреса, на который нужно выполнить
$i$
$i$-ю доставку. Доставки нужно выполнять строго в порядке символов строки
$s$
$s$.

Пронумеруем строки решётки от
1
1 до
$n$
$n$ сверху вниз, а столбцы — от
1
1 до
$m$
$m$ слева направо. Клетка
(
$x$
,
$y$
)
$(x,y)$ — пересечение строки
$x$
$x$ и столбца
$y$
$y$. Изначально ровер находится в клетке
(

s
x
,
s
y
)
(s
x

,s
y

). В клетке
(
i
,
j
)
(i,j) расположен адрес типа
x
i
,
j
x
i,j

. Поскольку речь идёт о типах адресов, для каждого типа таких адресов на карте может быть много — ровер может выполнить доставку для требуемого типа в любой клетке с этим типом. Перемещаться ровер может только между соседними по стороне клетками; каждый такой переход занимает ровно одну единицу времени. Передача заказа, в случае когда ровер находится в нужной клетке, считается мгновенной.

Помогите определить, за какое минимальное время ровер сможет выполнить все доставки из списка
s
s.

В первой строке даны два целых числа
n
n и
m

m — размеры города (
1
≤
n
,
m
≤
300
1≤n,m≤300). Во второй строке даны два целых числа
s
x
s
x

и
s
y
s
y

— начальные координаты ровера (
1
≤
s
x
≤
n
1≤s
x

≤n,
1
≤
s
y
≤
m
1≤s
y

≤m).

Каждая из следующих
$n$
$n$ строк состоит ровно из
$m$
$m$ строчных английских букв. В
$i$
$i$-й из этих строк
$j$
$j$-й символ задаёт
$x_{i,j}$
 — тип адреса в клетке
$(i,j)$. Гарантируется, что каждый из
$26$
$26$ типов встречается хотя бы в одной клетке.

Далее вводится строка
$s$
$s$ из строчных английских букв — последовательность доставок (
$1 \le |s| \le 300$
$1 \le |s| \le 300$).

Выведите единственное число — минимальное время, необходимое роверу для выполнения всех доставок из списка

s
s.

В первом примере оптимальный маршрут: дойти за
12
12 шагов до ближайшей клетки с адресом типа ‘n’, затем спуститься на
1
1 клетку вниз и последовательно выполнить доставки к адресам типов ‘u’ и ‘t’, стоящим подряд справа, что потребует ещё
4
4 шага.

Во втором примере оптимальный маршрут задаётся точками
(4,4), ‘s’
(5,5), ‘q’
(3,5), ‘u’
(5,3), ‘i’
(6,4), ‘r’
(4,5) (дважды), ‘e’
(4,6) и ‘l’
(6,7).

Пример 1
Ввод
Вывод
2 26
1 1
abcdefghijklmnopqrstuvwxyz
abtxyzutalkhfdyutxzbzhhawj
nut
17
Пример 2
Ввод
Вывод
7 7
4 4
abcdefg
xyzabch
wnopqdi
vmvwrej
ulutsfk

tkjihgl
srqponm
squirrel
17

**РЕШЕНИЕ**

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m;
    if (!(cin >> n >> m)) return 0;
    int sx, sy; cin >> sx >> sy; --sx; --sy;
    vector<string> g(n);
    for (int i = 0; i < n; ++i) cin >> g[i];
    string s; cin >> s;
    string t; t.reserve(s.size());
    for (char c : s) if (t.empty() || t.back() != c) t.push_back(c);

    const int INF = 1e9;
    vector<int> dp(n * m, INF), nxt(n * m, INF);
    dp[sx * m + sy] = 0;

    auto transformL1 = [&](vector<int>& a) {
        for (int i = 0; i < n; ++i) {
            int base = i * m;
            for (int j = 1; j < m; ++j) a[base + j] = min(a[base + j], a[base + j - 1] + 1);
            for (int j = m - 2; j >= 0; --j) a[base + j] = min(a[base + j], a[base + j + 1] + 1);
        }
        for (int j = 0; j < m; ++j) {
            for (int i = 1; i < n; ++i) {
                int idx = i * m + j, up = (i - 1) * m + j;
                int v = a[up] + 1; if (a[idx] > v) a[idx] = v;
            }
            for (int i = n - 2; i >= 0; --i) {
                int idx = i * m + j, dn = (i + 1) * m + j;
                int v = a[dn] + 1; if (a[idx] > v) a[idx] = v;
            }
```

```
        }
    };

    transformL1(dp);
    fill(nxt.begin(), nxt.end(), INF);
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            if (g[i][j] == t[0]) nxt[i * m + j] = dp[i * m + j];
    dp.swap(nxt);

    for (int k = 1; k < (int)t.size(); ++k) {
        transformL1(dp);
        fill(nxt.begin(), nxt.end(), INF);
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < m; ++j)
                if (g[i][j] == t[k]) nxt[i * m + j] = dp[i * m + j];
        dp.swap(nxt);
    }

    int ans = INF;
    for (int v : dp) if (v < ans) ans = v;
    cout << ans << "\n";
    return 0;
}
```

## D. Распределённая система (100p)

Полное решение

Ограничение времени 1 секунда

Ограничение памяти 64 Мб

Ввод стандартный ввод или input.txt

Вывод стандартный вывод или output.txt

Рассмотрим распределённую вычислительную систему, состоящую из

n

n вычислительных узлов (серверов), между которыми установлены каналы связи, образующие дерево (т.е. сеть без циклов, где между любой парой узлов существует единственный путь). Эта структура определяет топологию обмена сообщениями между узлами — какие узлы могут напрямую взаимодействовать при выполнении распределённого алгоритма.

Также имеется множество из

n

n географически распределённых дата-центров, каждый из которых может быть назначен для размещения одного из вычислительных узлов. Эти дата-центры расположены в точках общего положения на карте (никакие три не лежат на одной прямой и никакие два не совпадают).

Задача: необходимо сопоставить каждому вычислительному узлу (вершине дерева) уникальный дата-центр (точку на плоскости), так, чтобы если визуализировать каналы связи между узлами как прямые линии между соответствующими дата-центрами, то любые несмежные каналы не пересекались. Это обеспечивает, например, физическую или логическую изоляцию между несвязанными напрямую частями распределённой системы — что важно для устойчивости и безопасности ML-систем.

Формат ввода
Первая строка содержит одно целое число
t
t — количество тестов.
1
≤
t
≤
1000
1≤t≤1000

Далее для каждого теста:

В первой строке теста содержится одно целое число
n
n — количество узлов (и одновременно точек).
2
≤
n
·
t
≤
1000
2≤n·t≤1000

Следующие
n
–
1

n−1 строк содержат описание рёбер дерева. Каждая строка содержит два целых числа
u
u и
v
v (
1
≤
u
,
v
≤
n
1≤u,v≤n) — номера вершин, соединённых ребром. Дерево не содержит циклов и состоит из ровно
n
−
1
n−1 ребра.

Далее идут
n
n строк, каждая из которых содержит два действительных числа
x
i
x
i

и
y
i
y
i

— координаты
i
i-й точки на плоскости, в которую можно сопоставить вершину.
−
1
0
4
<
x

i
,
y
i
<
1
0
4
−10
4
<x
i

,y
i

<10
4

Формат вывода
Для каждого теста выведите одну строку из
n
n целых чисел
p
1
,
p
2
,
...
,
p
n
p
1

,p
2

,...,p

n

— перестановку чисел от 1 до
n
n, где
p
i
p
i

означает, что вершине
i
i дерева сопоставляется
p
i
p
i

-я точка из входного списка.

Таким образом, для каждой вершины дерева выбирается уникальная точка на плоскости, и соединение вершин прямыми отрезками между сопоставленными точками не приводит к пересечению несмежных рёбер.

Пример 1
Ввод
Вывод
1
2
1 2
292.365297 561.624168
28.742075 869.836531
2 1
Пример 2
Ввод
Вывод
1
3
2 1
1 3
38.082299 -593.978441

-42.281845 -296.136429
217.814614 -47.946068
3 2 1
Примечания
Для дебага решений предлагается использовать написанный визуализатор.

```
import argparse
import matplotlib.pyplot as plt



def read_input_and_output(infile, outfile):
    # Чтение входа
    input_data = list(map(float, infile.read().split()))
    it = iter(input_data)
    t = int(next(it))
    if t != 1:
        raise ValueError("Визуализатор поддерживает только один тест.")
    n = int(next(it))
    edges = [(int(next(it)) - 1, int(next(it)) - 1) for _ in range(n - 1)]
    points = [(next(it), next(it)) for _ in range(n)]

    # Чтение вывода (перестановка)
    perm = list(map(int, outfile.read().split()))
    if len(perm) != n:
        raise ValueError("Размер перестановки не совпадает с числом вершин")

    assigned = [points[p - 1] for p in perm]
    return n, edges, assigned



def visualize(n, edges, assigned, show_ids=False):
    fig, ax = plt.subplots()
    xs, ys = zip(*assigned)
    ax.scatter(xs, ys, color="blue")

    for i, (x, y) in enumerate(assigned):
        if show_ids:
            ax.text(x, y, str(i + 1), fontsize=8, ha="right", va="bottom")

    for u, v in edges:
        x1, y1 = assigned[u]
```

```
            x2, y2 = assigned[v]
            ax.plot([x1, x2], [y1, y2], color="black")

    ax.set_aspect("equal")
    ax.set_title("Дерево на плоскости")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.tight_layout()
    plt.show()


def main():
    parser = argparse.ArgumentParser(description="Визуализатор вложенного дерева")
    parser.add_argument(
        "--infile",
        type=argparse.FileType("r"),
        help="Входной файл",
        default="input.txt",
    )
    parser.add_argument(
        "--outfile",
        type=argparse.FileType("r"),
        help="Файл с перестановкой",
        default="output.txt",
    )
    parser.add_argument(
        "--show-ids", action="store_true", help="Показывать номера вершин", default=True
    )
    args, unknown = parser.parse_known_args()

    n, edges, assigned = read_input_and_output(args.infile, args.outfile)
    visualize(n, edges, assigned, args.show_ids)


if __name__ == "__main__":
    main()
```

**РЕШЕНИЕ**

```
#include <bits/stdc++.h>
using namespace std;
```

```
struct P { double x, y; };

int n;
vector<vector<int>> g;
vector<int> sz, assign_idx;
vector<P> pts;

void dfs_size(int u, int p){
    sz[u] = 1;
    for(int v: g[u]) if(v!=p){ dfs_size(v,u); sz[u]+=sz[v]; }
}

double ang(const P& a, const P& o){
    return atan2(a.y - o.y, a.x - o.x);
}

void solve(int u, int p, vector<int> vec){
    int best = 0;
    for(int i=1;i<(int)vec.size();++i){
        int a = vec[i], b = vec[best];
        if (pts[a].x < pts[b].x || (pts[a].x == pts[b].x && pts[a].y < pts[b].y)) best = i;
    }
    swap(vec[0], vec[best]);
    int pu = vec[0];
    assign_idx[u] = pu;

    vector<int> rest(vec.begin()+1, vec.end());
    sort(rest.begin(), rest.end(), [&](int i, int j){
        double ai = ang(pts[i], pts[pu]), aj = ang(pts[j], pts[pu]);
        return ai < aj;
    });

    int off = 0;
    for(int v: g[u]) if(v!=p){
        vector<int> block(rest.begin()+off, rest.begin()+off+sz[v]);
        off += sz[v];
        solve(v, u, block);
    }
}
```

```
int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t;
    if(!(cin>>t)) return 0;
    while(t--){
        cin>>n;
        g.assign(n, {});
        for(int i=0;i<n-1;++i){
            int u,v; cin>>u>>v; --u;--v;
            g[u].push_back(v); g[v].push_back(u);
        }
        pts.resize(n);
        for(int i=0;i<n;++i) cin>>pts[i].x>>pts[i].y;

        sz.assign(n,0);
        dfs_size(0,-1);

        vector<int> all(n);
        iota(all.begin(), all.end(), 0);

        assign_idx.assign(n,-1);
        solve(0,-1,all);

        for(int i=0;i<n;++i){
            if(i) cout<<' ';
            cout<<assign_idx[i]+1;
        }
        cout<<"\n";
    }
    return 0;
}
```

## A _ LLM Scaling Week 1 (10p/10 ‘ 9 commits)

Условие

Ваша задача - ускорить forward pass функции swiglu:

```
def swiglu(a, b):
    return torch.nn.functional.silu(a) * b
```

Ваша имплементация должна быть написана на Triton, вызываться через интерфейс torch.ops.llm_scaling_week.swiglu_fwd(a, b) и возвращать только один torch.Tensor - результат операции.

Имплементация будет проверяться на корректность и производительность. Для прохождения теста на корректность результат вашей функций должен совпадать на torch.allclose с выходом eager-имплементации swiglu. Для прохождения теста на производительность ваша функция должна занимать

≤

75

≤75% от скорости eager-имплементации swiglu.

Гарантируется, что на вход будут поданы contiguous-тензоры. Типы и размерности входных тензоров a и b совпадают. Обратите внимание, что функция должна уметь работать как с fp32, так и с bf16 тензорами. Функция должна уметь работать эффективно вне зависимости от размерности тензоров.

Референсное решение проходит все тесты и на H100, и в Google Colab.

Ограничение по времени:

Ваше решение должно проходить тесты быстрее, чем за 1 минуту.

Система оценивания

Количество баллов за задачу: 10

Ограничения на количество посылок

Максимальное количество посылок по задаче: 10

Максимальное количество посылок по задаче за скользящие 24 часа: 2

Примечание

Логи тестирования можно посмотреть, скачав вывод в тесте 1. Не переименовывайте файл solution.py. Ваше решение должно быть в этом файле.

**Решение** solution.py

## B _ LLM Scaling Week 2 (15p/15 ‘ 4 commits)

Условие задачи находится в ноутбуке внутри репозитория.

Ограничение по времени:

Ваше решение должно проходить тесты быстрее, чем за 8 минут.

Система оценивания
Количество баллов за задачу: 15

Ограничения на количество посылок

Максимальное количество посылок по задаче: 10
Максимальное количество посылок по задаче за скользящие 24 часа: 2
Примечание
Важно: после того, как вы решите задачу, пожалуйста, перенесите реализацию функций: quantize_int8_perrow_kernel, quantize_int8, perrow_w8a8_matmul_kernel, matmul_quantize_int8 в файл solution.py. Внутри этого файла уже определены сигнатуры этих функций, вставьте реализацию вместо pass. Не переименовывайте файл, не удаляйте, не добавляйте функции в файл.

Логи тестирования можно посмотреть, скачав вывод в тесте 1.

**FULL SOL FILE (.ipynb)** task.ipynb

**РЕШЕНИЕ** solution.py

## C _ LLM Scaling Week 3 (25p/25 ‘ 14 commits)

Ваша задача - написать эффективную имплементацию операцию padded_moe_permute.

Ваша функция должна называться submission и иметь следующую сигнатуру:

```
def submission(
```

```
    x: torch.Tensor,  # (num_tokens, hidden_size) - входной тензор токенов, каждый размерности hidden_size
    top_experts: torch.Tensor,  # (num_tokens, topk) - для каждого токена указано topk экспертов, которые он активирует
    tokens_per_expert: torch.Tensor, # (num_experts,) - тензор размерности числа экспертов, i-ый элемент - сколько токенов приходит в i-ого эксперта
    topk: int,  # сколько экспертов активируются на каждый токен, например, 8
    num_experts: int,  # сколько всего экспертов в MoE, например, 128
) -> tuple[
    torch.Tensor,  # (max_padded, hidden_size) - padded_tokens, результат пермьюта с паддингами
    torch.Tensor # (num_experts,) - padded_tokens_per_expert, сколько токенов приходят в каждого эксперта вместе с паддингами
]
```

Для начала рассмотрим стандартную функцию moe_permute без учета паддингов:
На вход permute-функции приходит тензор размерности (num_tokens, hidden_size). Обычно в MoE пермьют переставляет токены так, чтобы токены, попадающие в одного эксперта, находились друг за другом. Например, путь на вход подается

```
x = tensor([[-0.0236, -0.5368, -0.5663],
        [ 0.7778, -0.8583, -0.1123],
        [ 0.1981, -0.3514, -0.9443],
        [-2.0655, -0.9424,  0.9870]])
top_experts = tensor([[1, 3],  # токен 0 выбирает экспертов 1 и 3
                [2, 5],    # токен 1 выбирает 2 и 5
                [3, 5],    # токен 2 выбирает 3 и 5
                [2, 4]])   # токен 3 выбирает 2 и 4
```

В данном случае topk=2, каждый токен выбирает 2 экспертов. Выходной тензор будет иметь размерность (num_tokens * topk, hidden_size), там сначала будут записаны токены для 0ого эксперта, потом для 1ого, потом для 2ого и так далее. В данном случае:

```
out = tensor([[-0.0236, -0.5368, -0.5663],# токен 0 -> эксперт 1
        [ 0.7778, -0.8583, -0.1123],  # токен 1 -> эксперт 2
        [-2.0655, -0.9424,  0.9870],  # токен 3 -> эксперт 2
        [-0.0236, -0.5368, -0.5663],  # токен 0 -> эксперт 3
        [ 0.1981, -0.3514, -0.9443],  # токен 2 -> эксперт 3
        [-2.0655, -0.9424,  0.9870],  # токен 3 -> эксперт 4
        [ 0.7778, -0.8583, -0.1123],  # токен 1 -> эксперт 5
        [ 0.1981, -0.3514, -0.9443]]) # токен 2 -> эксперт 5
```

Тензор размерности (num_experts,), который показывает, сколько токенов идут в каждого эксперта, назовем батч сайзами. В примере выше батч сайзы - это [1, 2, 2, 1, 2].

Теперь к нашей задаче
При использовании FP8-умножения из DeepGEMM (и других современных кернелов) часто ожидается TMA-алаймент тензора, то есть появляется требование делимости размерностей на 128. Это нужно для использования Tensor Memory Accelerator-а на H100 для асинхронного копирования из памяти.

В случае moe_permute это означает необходимость делимости батч сайзов на 128, то есть чтобы в каждого эксперта приходило делящееся на 128 число токенов. В примере выше батч сайзы [1, 2, 2, 1, 2] станут [128, 128, 128, 128, 128]. А, например, [128, 1, 129] перейдет в [128, 128, 256].

Чтобы добиться такой гарантии, нам придется западдить результат пермьюта. Теперь он не обязательно будет иметь размерность num_tokens * topk, а может содержать дополнительные нулевые токены. Ваша задача - написать функцию, которая будет делать то же самое, что и обычный moe_permute, но уже с паддингами - то есть дополнительно будет гарантировать, что первый токен для каждого эксперта начинается с индекса, делящегося на 128.

В случае входа из примера выше на выходе мы должны получить

```
tensor([[-0.0236, -0.5368, -0.5663], # токен 0 -> эксперт 1
        [ 0.0000,  0.0000,  0.0000],
        [ 0.0000,  0.0000,  0.0000],
        ...,
        [ 0.7778, -0.8583, -0.1123],  # токен 1 -> эксперт 2, индекс 128
        [-2.0655, -0.9424,  0.9870],  # токен 3 -> эксперт 2, индекс 129
        [ 0.0000,  0.0000,  0.0000],
        [ 0.0000,  0.0000,  0.0000],
        [ 0.0000,  0.0000,  0.0000]
        ...,
        [ 0.0000,  0.0000,  0.0000]])
```

Для имплементация можно использовать как Triton, так и обычный Torch.

Имплементация будет проверяться на корректность и производительность. Для прохождения теста на корректность результат вашей функций должен совпадать на torch.allclose с выходом eager-имплементации. Для прохождения теста на производительность ваша функция должна выдавать скорость примерно совпадающую с нашей референсной имлпементацией. Наша референсная имплементация не очень эффективная, поэтому не спешите сразу начинать с Triton.

Для простоты вам также дан неэффективный код с for-ами.

Ограничение по времени:

Ваше решение должно проходить тесты быстрее, чем за 3 минуты.

Система оценивания
Количество баллов за задачу: 25

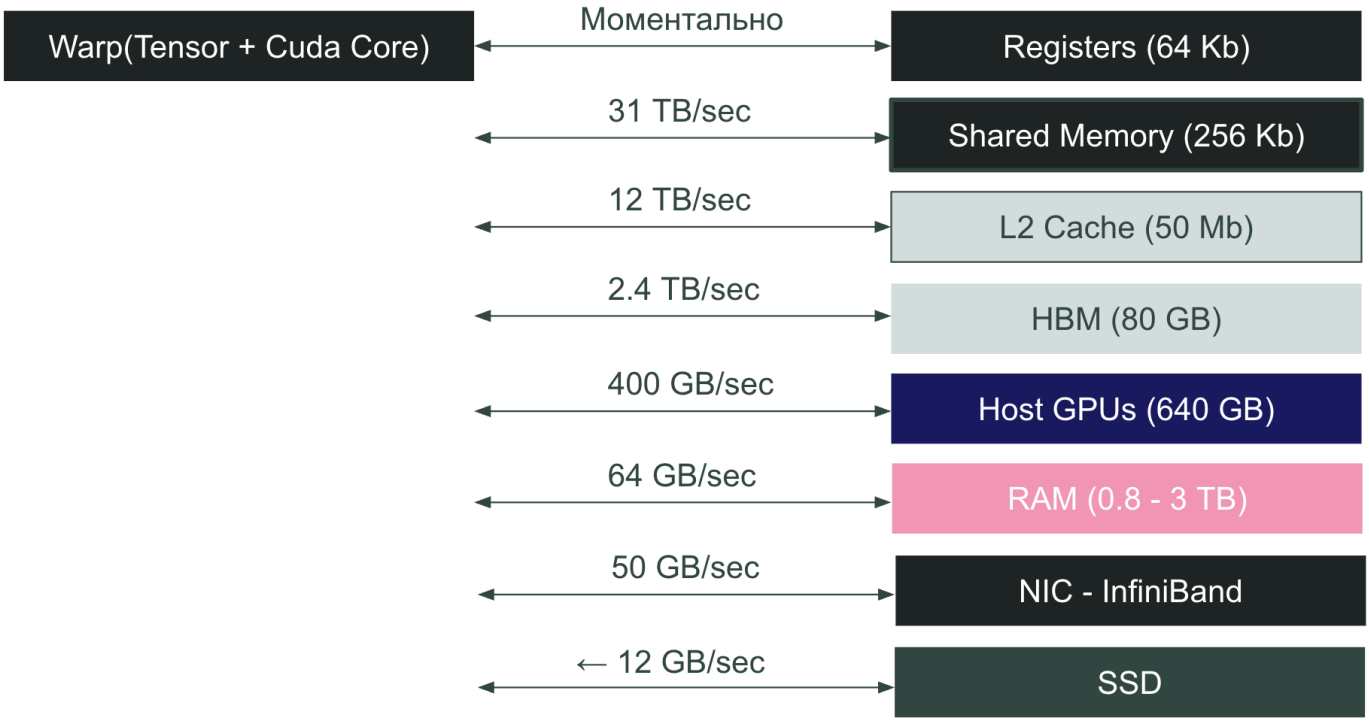
Ограничения на количество посылок

Максимальное количество посылок по задаче: 10
Максимальное количество посылок по задаче за скользящие 24 часа: 2
Примечание
Логи тестирования можно посмотреть, скачав вывод в тесте 1. Не переименовывайте файл solution.py. Ваше решение должно быть в этом файле.

Так же в solution.py закралась синтаксическая ошибка в строчку 68. Нужно будет убрать в своем решении точку после torch.Tensor.

**РЕШЕНИЕ** solution.py

Вопрос 1
Используя данную схему и материалы с курса, оцените, сколько времени при идеальных коммуникациях должна занимать сборка в FSDP модели с 235B параметрами, где каждый занимает 2 байта, а параметры шардируются на всех GPU кластера.




Примечание
Ответ введите десятичным числом с разделителем точка с точностью до двух знаков после запятой, единицы измерения секунды.

ОТВЕТ 1.18

Вопрос 2
Оцените, сколько времени в секундах при оптимальных коммуникациях займет Upload из CPU RAM в GPU RAM модели в 50GB активаций?

Примечание
Ответ введите десятичным числом с разделителем точка с точностью до двух знаков после запятой.

ОТВЕТ 0.78

Вопрос 3
Скорось HBM - 2.4 TB/sec, скорость GPU в Tensor Core - 800 TFLOPS. Оцените время работы функции forward в мкрсек (при минимальном времени запуска одного матричного умножения в 10 мкрсек):

```
q = Linear(4096, 8192, device='cuda', dtype=torch.bfloat16)
k = Linear(4096, 512, device='cuda', dtype=torch.bfloat16)
v = Linear(4096, 512, device='cuda', dtype=torch.bfloat16)

x = torch.randn(512, 4096, device='cuda', dtype=torch.bfloat16)

def forward(x):
 return q(x), k(x), v(x)

forward(x)
```

Примечание

Ответ введите натуральным числом, единицы измерения: мкрсек.

ОТВЕТ 63

Вопрос 4

Скорось HBM - 2.4 TB/sec, скорость GPU в Tensor Core - 800 TFLOPS. Оцените время работы функции forward в мкрсек (при минимальном времени запуска одного матричного умножения в 10 мкрсек):

```
qkv = Linear(4096, 8192 + 2 * 512, device='cuda', dtype=torch.bfloat16)

 x = torch.randn(512, 4096, device='cuda', dtype=torch.bfloat16)

 def forward(x):
 return qkv(x)
 forward(x)
```

Примечание

Ответ введите натуральным числом, единицы измерения: мкрсек.

ОТВЕТ 48

Вопрос 5

Скорось HBM - 2.4 TB/sec, скорость GPU в Tensor Core - 800 TFLOPS. Оцените время forward:

```
x = torch.randn(64, 8192, 8192, device='cuda', dtype=torch.bfloat16)
d = torch.sqrt(128)

def forward(x, d):
 return x / d
forward(x, d)
```

Примечание
Ответ введите натуральным числом, единицы измерения: мкрсек.

ОТВЕТ 7158

Вопрос 6
Скорось HBM - 2.4 TB/sec, скорость GPU в Tensor Core - 800 TFLOPS. Оцените время forward при оптимальной реализации ядра attention (Flash Attention):

```
q_seqlen = 16
kv_seq_len = 8192
num_heads = 64
head_dim = 128

q = torch.randn(q_seqlen, num_heads, head_dim, device='cuda', dtype=torch.bfloat16)
k = torch.randn(kv_seq_len, num_heads, head_dim, device='cuda', dtype=torch.bfloat16)
v = torch.randn(kv_seq_len, num_heads, head_dim, device='cuda', dtype=torch.bfloat16)

out = attention(q, k, v)
```

Примечание
Ответ введите натуральным числом, единицы измерения: мкрсек.

ОТВЕТ 112

Вопрос 7

Оптимальная реализация attention выше является:
Выберите вариант ответа

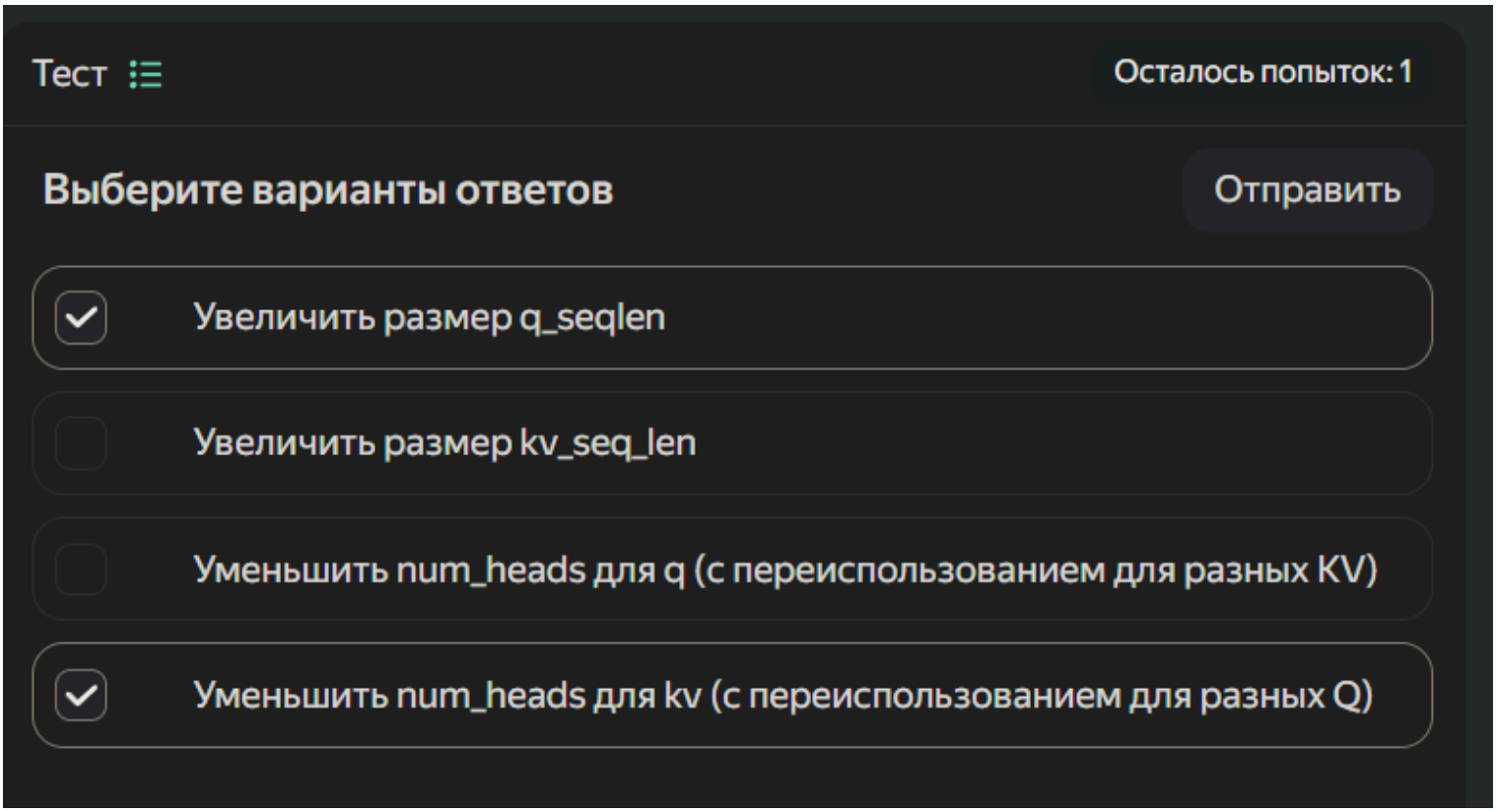
Compute bound

Memory bound

Communication bound

ОТВЕТ Memory bound

Вопрос 8
Что может повысить утилизацию attention выше?




Вопрос 9
Сколько SMок (Streaming Multiprocessor-ов) можно найти в H100 SXM, о которой шла речь в Лекции 3?

Осталось попыток: 0
Выберите вариант ответа

132

24

520

4

ОТВЕТ 132

Вопрос 10
Какая скорость работы с HBM (High-Bandwidth Memory) заявлена Nvidia в H100 SXM?

Осталось попыток: 0
Выберите вариант ответа

3.35 TB/s

9 TB/s

15 TB/s

990 GB/s

ОТВЕТ 3.35 TB/s

Вопрос 11
Какой суммарной вместимостью обладает L1-cache/Shared Memory в одной H100 SXM?

Осталось попыток: 0
Выберите вариант ответа

33 MB

132 MB

80 GB

128 kB

ОТВЕТ 33 MB

Вопрос 12
Пусть мы хотим сложить на GPU два вектора в FP32 и произвести операцию
c
=
a
+
b
.
c=a+b.
В каждом векторе N элементов. Сколько байт будут пропущены через память (HBM)?

Осталось попыток: 0
Выберите вариант ответа

3 N

4 N

8 N

12 N

ОТВЕТ 12N

Вопрос 13
Пусть мы хотим сложить на GPU два вектора в FP32 и произвести операцию
c
=

a
+
b
.
c=a+b.
В каждом векторе
N
N элементов. Оцените Arithmetic intensity этой операции

Осталось попыток: 0
Выберите вариант ответа

1/12

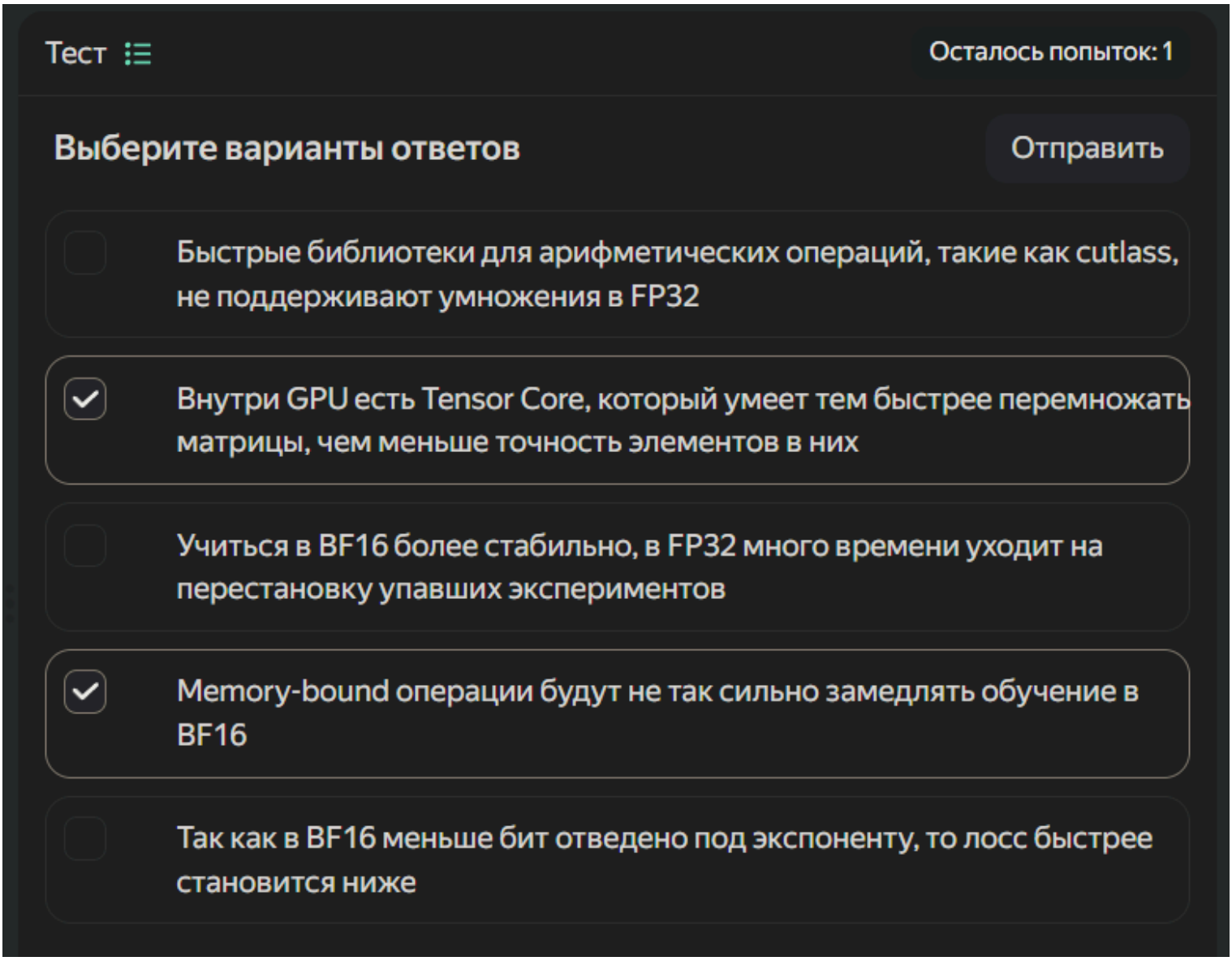
1/2

1

2/3

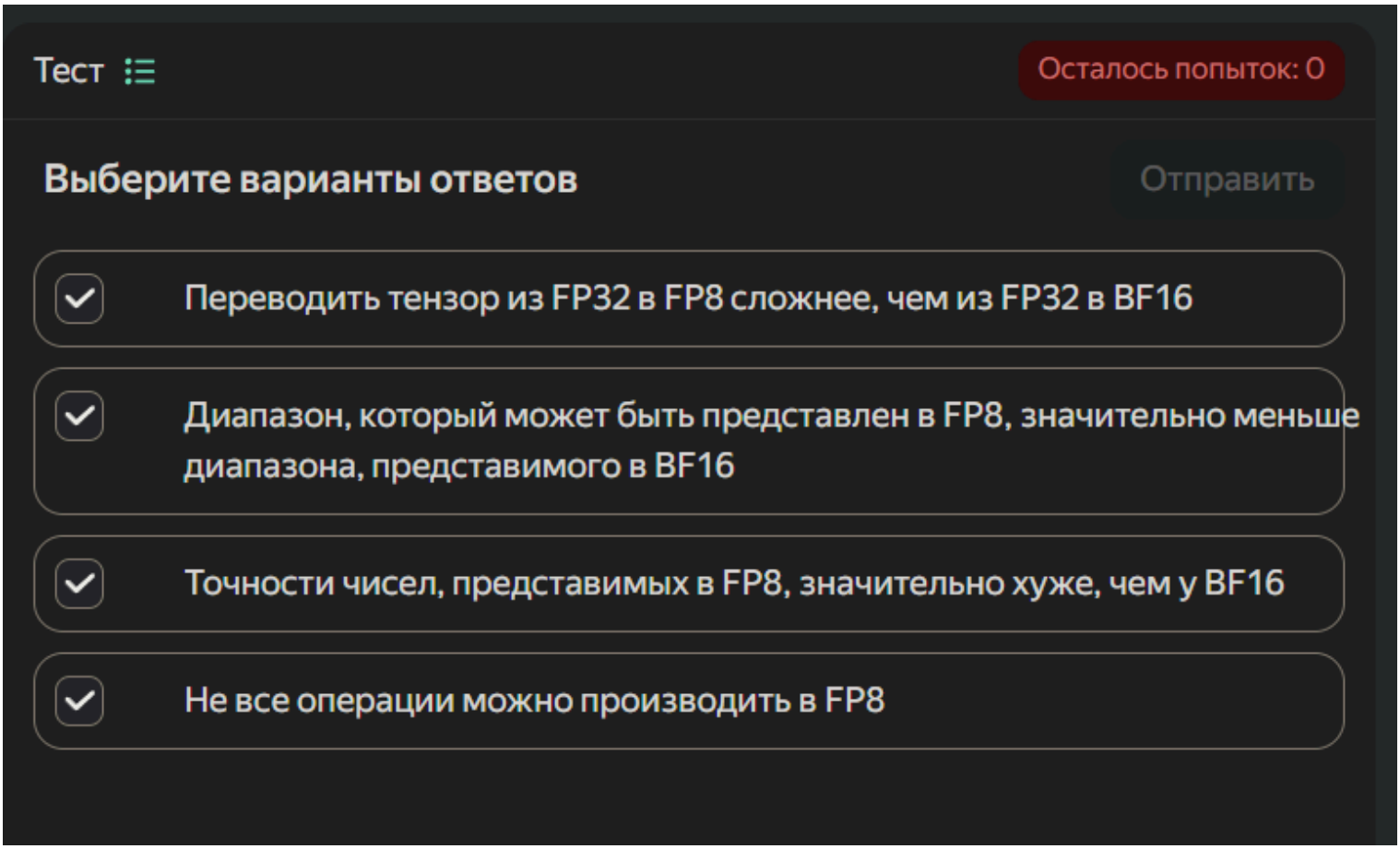
ОТВЕТ 1/12

Вопрос 14
Почему обучение в BF16 быстрее, чем обучение в FP32? Выберите все варианты.



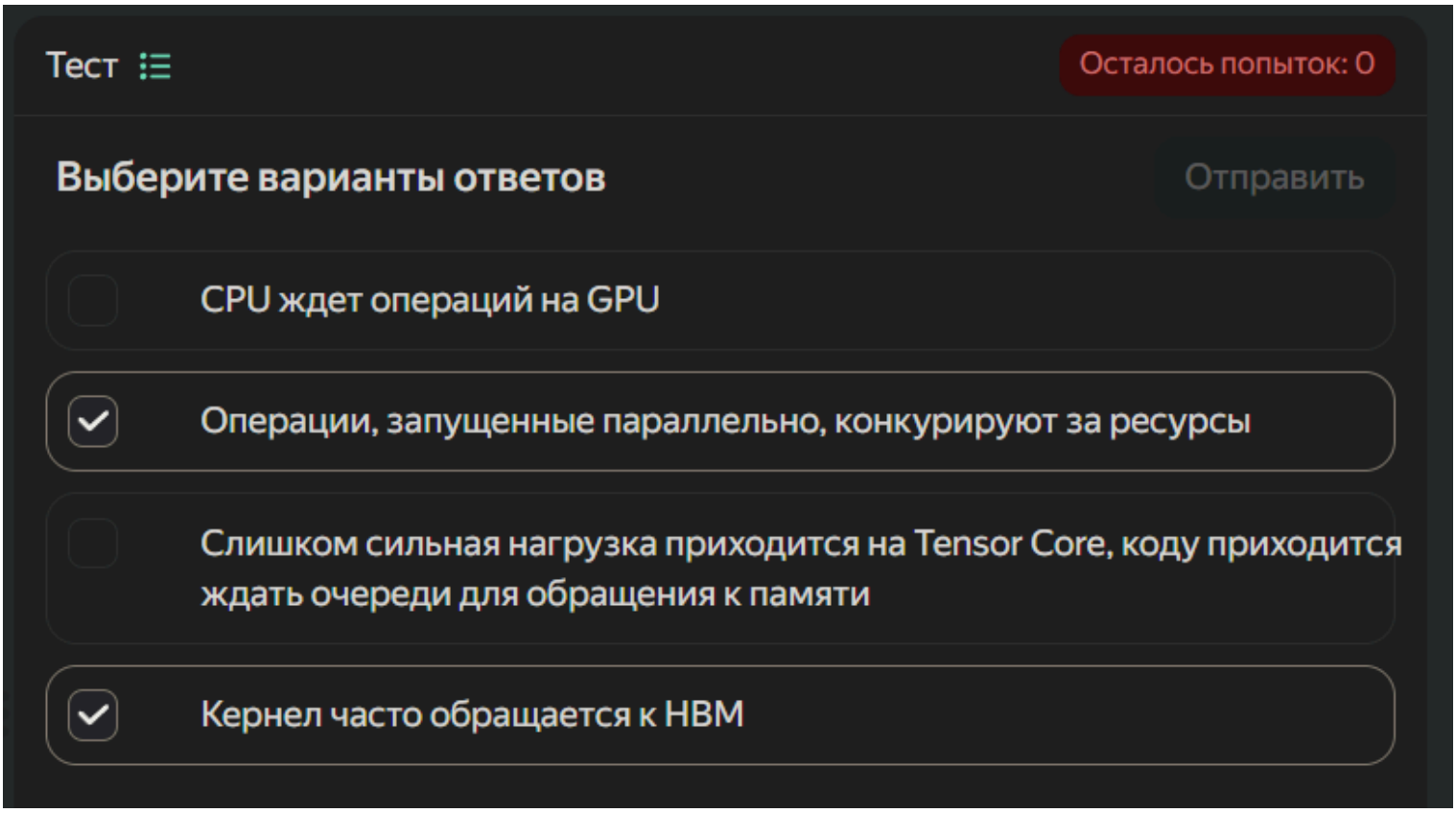

Вопрос 15

В чем сложность обучения в FP8 в сравнении с BF16? Выберите все варианты.




Вопрос 16 (**НЕПРАВИЛЬНО**)

Выберите причины, по которым может замедляться обучение на GPU-кластере. Укажите все варианты.




Вопрос 17(**НЕПРАВИЛЬНО**)
Пользуясь swiglu и swiglu_compiled измерьте время исполнения этих функций. Чему равно time_compiled / time_eager?

```
import torch
import triton
import torch.nn.functional as F

def swiglu(a, b):
    return F.silu(a) * b

swiglu_compiled = pass # TODO: your code here
a = torch.randn(128, 128, device="cuda")
b = torch.randn(128, 128, device="cuda")

out = swiglu_compiled(a, b)

time_eager = TODO_BENCH(swiglu(a, b))
time_compiled = TODO_BENCH(swiglu_compiled(a, b))
time_compiled / time_eager
```

Осталось попыток: 0
Выберите вариант ответа

0.3

0.7

1

1.5

Вопрос 18 (**НЕПРАВИЛЬНО**)
Какая операция работает быстрее на 128 хостах из 8 H100 GPU (nvlink/infiniband) с настройками по умолчанию, если вход - тензор из одного fp32 элемента

Осталось попыток: 0
Выберите вариант ответа

torch.distributed.all_gather_into_tensor(output_tensor, input_tensor)

torch.distributed.all_reduce(tensor)

torch.distributed.reduce(tensor, dst=0)

Вопрос 19 (**НЕПРАВИЛЬНО**)
Используя значения эффективных пропускных способностей с этого графика, рассчитайте, сколько времени займет операция reduce_scatter на 4096 GPU (512 нод) при использовании HSDP, где размер Sharding Group - 128 GPU, а Replicating Group - 32 GPU, если размер входного тензора - 512 MiB.

Округляйте пропускную способность с графиков до числа кратного 50 GB/s

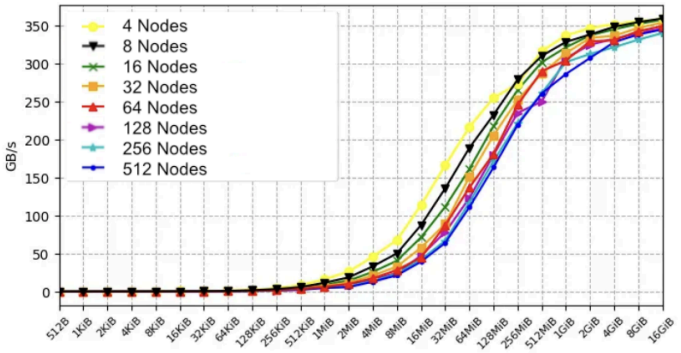
Считайте, что пропускная способность reduce_scatter совпадает с пропускной способностью all_gather

Считайте, что если у вас есть 8 независимых пересылок по <M> байт на <N> хостов, где в каждой пересылке из каждого хоста участвует ровно 1 GPU, то это эквивалентно соответсвует точке "8 * <M> байт" на графике "<N> Nodes"

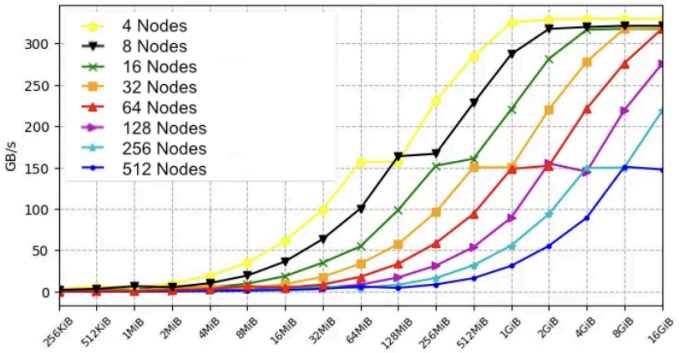
Считайте, что коммуникации по Sharding Group и Replicating Group не перекрываются

Учитывайте, что GiB - это 2^30 байт, а GB - 10^9 байт

Ответ дайте в милисекундах с точностью до десятых




**(a)** Bandwidth of NCCL **AllReduce** using a tree algorithm and scales well with number of nodes (i.e. higher bandwidth).




**(b)** Bandwidth of NCCL **AllGather** using ring algorithms; scales poorly with the number of nodes (i.e. lower bandwidth).

Ответы НЕПРАВИЛЬНЫЕ 2.5 3.6 9.0 8.9 40.0

Вопрос 20

В операции reduce_scatter с использованием алгоритма ring и входным bf16 тензором, в каком типе данных происходят пересылки частичных сумм?

Осталось попыток: 0

Выберите вариант ответа

bf16

fp32

ОТВЕТ bf16