

```
!nvcc -O3 task1_cuda_sum.cu -o task1
```

```
!./task1
```

TASK 1 – CUDA sum

CPU sum = 100000, time = 0.170962 ms

GPU sum = 0, time = 7.44035 ms

```
!nvcc -O3 task2_scan_shared.cu -o task2
```

```
!./task2
```

TASK 2 – Prefix sum (shared memory)

CPU time = 0.001367 ms

GPU time = 7.424 ms

Last element = 0

TASK 3 – Floyd-Warshall (Scatter + Allgather)

N = 256, processes = 1

Execution time: 0.0271204 seconds

Top-left 10x10 of distance matrix:

```
0 5 4 3 4 5 6 2 7 7
4 0 4 4 5 5 3 4 4 4
3 5 0 5 3 1 3 5 5 4
5 4 5 0 5 4 4 4 4 6
3 5 4 5 0 4 2 5 5 5
5 6 3 7 2 0 4 6 7 4
1 4 5 4 4 6 0 3 6 5
4 3 5 1 4 5 4 0 5 5
5 4 4 4 4 4 4 3 0 4
6 6 6 7 6 5 6 6 8 0
```

```
[32] 1 сек. !mpic++ -O3 task4_mpi.cpp -o task4
!mpirun --allow-run-as-root -np 1 ./task4

TASK 4 — MPI
Processes: 1
Sum = 1e+06
Time = 0.00136449 s

[33] 0 сек. !mpirun --allow-run-as-root --oversubscribe -np 4 ./task4

... TASK 4 — MPI
Processes: 4
Sum = 1e+06
Time = 0.000509037 s
```

Ответы на контрольные вопросы

1. В чём отличие гибридных вычислений от CPU/GPU-only?

Гибридные вычисления используют CPU и GPU одновременно, позволяя CPU выполнять управляющие и последовательные части, а GPU — массово-параллельные вычисления.

2. Для каких задач целесообразно распределять вычисления между CPU и GPU?

Для задач с большим объёмом данных и высокой степенью параллелизма: обработка массивов, линейная алгебра, численные методы.

3. Разница между синхронной и асинхронной передачей данных?

Синхронная блокирует CPU до завершения копирования, асинхронная позволяет перекрывать передачу данных с вычислениями.

4. Почему асинхронная передача повышает производительность?

Потому что уменьшает простои CPU и GPU, позволяя выполнять вычисления параллельно с копированием данных.

5. Основные MPI-функции распределения и сбора данных?

MPI_Scatter, MPI_Gather, MPI_Reduce, MPI_Allreduce, MPI_Bcast.

6. Как число MPI-процессов влияет на время выполнения?

До определённого момента ускоряет выполнение, затем коммуникационные накладные расходы начинают доминировать.

7. Что ограничивает масштабируемость распределённых программ?

Коммуникации, синхронизация, дисбаланс нагрузки, пропускная способность сети.

8. Когда распределённые вычисления оправданы, а когда нет?

Оправданы для больших задач и кластеров; неэффективны для малых объёмов данных и высоких накладных расходов.