5. Лабораторная работа №2 язык Julia

5.1. Предварительные сведения

В языке Julia реализованы различные механизмы параллельных и асинхронных вычислений. Все они перечисленны на странице документации под названием Parallel Computing (ссылка).

- *Асинхронные задачи (tasks) и сопрограммы (coroutines).* Это базовый механизм обеспечения асинхронности, который не гарантирует параллельности.
- *Мультипоточность* (multi-threading). Механизм параллельных вычислений на основе потоков, который лучше всего подходит для компьютеров с многоядерными процессорами. На данный момент реализованы лишь базовые средства: запуск потоков, запуск задачи на выполнение в потоке, ожидание завершения задачи с получением результата и без, механизм блокировки ресурса для предотвращения гонки данных, атомарные операции и параллельный цикл `for`.
- *Распределенное программирование* (distributed computing). Это реализация параллельности на основе процессов для кластеров и распределенных систем. Предоставляет собой набор модулей, которые реализуют разные механизмы и подходы. Например, есть модуль `MPI.jl`, реализующий стандарт `MPI`.
- *Параллельные вычисления на графических процессорах.* Большой набор сторонних модулей, реализующих запуск Julia-кода на графических процессорах (видеокартах).

Для наших задач подходит параллельность, основанная на многопоточности, о которой можно прочитать в официальной документации по ссылке. Сразу же следует отметить, что средства многопоточного программирования в Julia крайне скупы, что ясно видно из объема документации. Это можно объяснить несколькими факторами.

- Язык довольно новый и поддержка параллельности была добавлена не сразу и постоянно улучшается и доделывается. В будущем, скорее всего, будут добавлены дополнительные функции и макросы.
- Разработчики языка умышлено сконцентрировались лишь на самых базовых возможностях, а реализацию более высокоуровневых концепций оставили разработчикам сторонних библиотек.

В заданиях лабораторной работы упор делается на концепцию редуцирования (reduce и map-reduce). Теоретически, ее можно реализовать самостоятельно, используя встроенные в Julia средства. Однако, предлагается использовать следующие библиотеки:

- `Folds.jl` — модуль реализует параллельное редуцирование. В функциональном программировании термин fold можно считать синонимом редуцирования. Ссылки: репозиторий, документация.
- `FLoops.jl` — модуль реализует параллельные циклы с помощью макроса `@floop` и редукцию в них с помощью макроса `@reduce`. Ссылки https://github.com/JuliaFolds/FLoops.jl, https://juliafolds.github.io/FLoops.jl/dev/.

Могут пригодиться также и дополнительные модули.

- `OnlineStats` — модуль для статистики. Репозиторий.
- `BenchmarkTools` — замеры производительности. Репозиторий.

Все модули можно установить стандартным способом, выполнив команду `add Folds FLoops OnlineStats BenchmarkTools`.

5.2. Задание №1

Реализуйте параллельные версии суммирования, нахождения максимума и минимума для массива с большим количеством данных.

- Запустите программу для 1, 2, 3 ..., n потоков, где n — количество потоков процессора (должны были выяснить в предыдущей лабораторной).

- Программа должна давать корректные вычислительные результаты для любого количества потоков: вычислять правильную сумму, максимум и минимум. Для проверки сделайте тесты.
- При этом программа должна ускоряться, то есть при увеличении количества потоков она должна работать быстрее. Чтобы это проверить необходимо нарисовать графики времени выполнения, ускорения и эффективности.

5.3. Задание №2

Теперь используем редукцию суммирования не как игрушечную функцию для синтетических тестов, а сделаем что-то более полезное. В частности, вычислим значение определенного интеграла с помощью формулы трапеции.

$$\int_a^b f(x)\mathrm{d}x \approx \frac{h}{2}(f(a)+f(b)) + \sum_{i=1}^{n-1} f(x_i)\cdot h,\ x_i = a + ih.$$

Реализуйте эту функцию методом суммирования с редуцированием. Программа, так же как и предыдущая, должна давать корректный вычислительный результат и должна ускоряться. Для проверки используйте интеграл, который можно вычислить аналитически.