0.1 Методика исследования

Первым этапом была сборка вычислительного пакета GROMACS.[1] Для корректной сборки необходимо ознакомиться с минимальным перечнем библиотек и поддерживаемых инструкций вычислительного пакета. Данную информацию можно найти в документации GROMACS. В данной работе вычислительный пакет включал следующие библиотеки (Таб.0.1). По результатам исследования HPC-систем помимо основных задач проекта, можно будет оценить эффективность данной сборки.

Библиотека	Описание
librt.so.1	Содержит функции для работы с различными таймерами и событиями.
libcufft.so.10	Содержит реализацию FFT (Быстрое преобразование Фурье) для CUDA.
libgomp.so.1	Содержит реализацию OpenMP.
libmpi.so.40	Содержит реализацию протокола MPI для использования в OpenMP.
libstdc++.so.6	Содержит реализацию стандартной библиотеки C++
libm.so.6	Содержит математические функции
libgcc_s.so.1	Содержит реализацию стандартных функций поддержки GCC
libpthread.so.0	Содержит реализацию многопоточности POSIX.
libc.so.6	Содержит реализацию стандартной библиотеки C.
libopen-rte.so.40	Содержит реализацию runtime environment, используется в реализации сообщений между процессами
libutil. so.1	Содержит набор вспомогательных функций (util), которые могут быть использованы различными программами в Linux

Таблица 0.1: Библиотеки для сборки вычислительного пакета GROMACS

Одной из важных особенностей GROMACS является поддержка параллельного вычисления задач в стандартах OpenMP и MPI.

MPI - программный интерфейс (API) для обмена информацией между параллельно работающими процессами. MPI используется на системах распределенной памятью,где каждый параллельный процесс работает в своем собственном пространстве памяти в отрыве от других.

OpenMP - это стандарт программирования параллельных вычислений на системах с общей памятью, позволяющий автоматически поделить задачу на несколько исполнительных потоков, которые будут работать параллельно, и управлять его выполнением.

Вместе, MPI и OpenMP обеспечивают возможности для параллельного выполнения задач различной сложности на многопроцессорных и многоядерных системах.

Применение MPI и OpenMP позволяет значительно ускорить вычисления и сделать обработку данных более эффективной.[2]

Вторым этапом является компиляция бенчмарка. Для компиляции бенчмарка в рамках вычислительного пакета GROMACS необходимо три файла. Файл топологии системы (.top) - содержит информацию о типах атомов, связях, силовых полях и описание модели, файл параметров моделирования (.mdp), который содержит информацию о выборе методов и алгоритмов, параметрах симуляции и выводе результатов, а также файл (.gro), который задает стартовую конфигурацию системы и содержит информацию о координатах атомов ApoA1 в растворе. После подготовки данных файлов, файл расчет (.tpr) компилировался следующей командой:

```
gmx_mpi grompp -f run.mdp -c apoa1gro -p apoa1top -o run.tpr -maxwarn 1
```

Где ключи `-f, -c, -p, -o` указывают на файлы, описанные выше. Ключ `-maxwarn 1` задает число максимально допустимых предупреждений, при генерации .tpr-файла.Если количество предупреждений превышает заданное число, то генерация файла будет прервана,данный ключ обеспечит корректную компиляцию исполнительного файла. (Файлы для компиляции были взяты с `https://gitlab.com/nidkond/apoa1-bench`)

Третьим этапом является корректный подбор параметров запуска, для этого использовался запуск в пакетном режиме. Использование пакетного режима выполнения задач является одним из основных преимуществ кластерных систем, поскольку позволяет оптимизировать потребление ресурсов суперкомпьютера и сделать процесс работы более эффективным. Скрипт-файл содержит команды для загрузки необходимых модулей, настройки переменных среды, запуска программы с определенными параметрами и т.д. Он также содержать запросы на выделение ресурсов, таких как количество вычислительных ядер, объем оперативной памяти, время выполнения и т.д. Полезным дополнением к скрипту является возможность сохранения вывода и ошибок в файлы на диск, что позволяет более эффективно отслеживать и отлаживать выполняемые задачи. Общий синтаксис скрипта имеет следующий вид (Рис.0.1)

Здесь `#!/bin/bash` сообщает операционной системе, что данный скрипт является скриптом для командной оболочки Bash. Строки, начинающиеся с `#SBATCH`, содержат директивы для пакета SLURM, который управляет задачами на кластере. Они задают параметры, такие как время выполнения, требуемые ресурсы (cores, memory, и др.), название задачи, лог файлы, и др. Далее идет блок, начинающийся со строки `module load`, в котором производится загрузка необходимых модулей. Затем можно перечислить блок настройки переменных или констант для выполнения задачи на

```
#! /bin/bash
#SBATCH <sbatch option>= <value>
...
#SBATCH <sbatch option>= <value>
…
module load <module_name>
...
export <environment variable>=<value>
…
<user commands>
```

Рис. 0.1: Общий синтаксис скрипт файла

кластере при помощи команды `export`. Наконец, блок `<пользовательские команды>` содержит команды, которые должны быть выполнены в рамках задачи на кластере. В рамках данного направления шаблон скрипт файла имел следующий вид (Рис.0.2):

```
#SBATCH --job-name= gromacs_<ID>        # Название задачи
#SBATCH -error= gromacs_<ID>-%j.err    # Файл для вывода ошибок
#SBATCH --output= gromacs_<ID>-%j.log # Файл для вывода результатов
#SBATCH --time={01:00:00}              # Максимальное время выполнения
#SBATCH --ntasks=<значение>            # Количество MPI процессов
#SBATCH --nodes=<значение>             # Требуемое кол-во узлов
#SBATCH --gpus=<значение>              # Требуемое кол-во GPU
#SBATCH --cores-per-socket=<значение> # Требуемое кол-во cores на сокет
#SBATCH --cpus-per-task=<значение>    # Требуемое кол-во CPU
#SBATCH --ntasks-per-node=1            # Требуемое кол-во процессов на узел
#SBATCH --constraint=<значение>      # Предпочтительный тип узлов

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK #Кол-во потоков OpenMP на task

module purge
module load GROMACS/2022.2             # Загрузка модуля

srun -np <> --mpi=pmix_v3 gmx_mpi mdrun -ntomp <Y> -v -s run.tpr -pin on
                                       # Выполнение расчёта
```

Рис. 0.2: Шаблон скрипт файла, используемого при запусках

С помощью данной конфигурации можно получить необходимые ресурсы и обеспечить оптимальные условия запуска на суперкомпьютере бенчмарка ApoA1,реализованного в рамках вычислительного пакета GROMACS.

Следующим шагом является определение компонент HPC-систем и определение набора инструментов для сбора данных.Для определения компонент суперкопьютера и их конфигурации на базе LINUX, можно использовать следующие команды: `scontrol`

`show node` — команда Linux, которая отображает информацию о всех узлах вычислительного кластера, подключенных к системе управления кластером Slurm.Более детальную информацию о центральных процессоров можно получить с помощью команды: `cat /proc/cpuinfo`.Из результатов данных команд были сделаны выводы о конфигурации каждого узла, которые использовались для определения параметров запуска. Для сбора данных об использовании ресурсов системы и анализа производительности использовались следующие методы:

1.Встроенные в GROMACS профилировщики. Из них можно получить детальную информацию о выделенных компонентах суперкомпьютера для запуска бенчмарка, количество операций с плавающей точкой (FLOP) для всех видов взаимодействий,общее количество FLOP бенчмарка. Информацию о использовании вычислительных ресурсов для различных частей программного обеспечения.

2.Инструмент профилирования для Linux-Perf.[3] С помощью данного инструмента можно получить более детальную информацию о результатах запуска бенчмарка, такие как количестве циклов процессора в ходе выполнения задачи, количестве инструкций, средней частоте процессора, кэш-промахах и других параметров. Она может быть использована для поиска "узких"мест и оптимизации производительности.

3.Для динамического анализа и контроля запуска использовалась интерактивная утилита Linux htop и утилита командной строки scontrol show <JOBID>

4.HPC TaskMaster - встроенная система мониторинга эффективности задач суперкомпьютера "cHARISMa". С помощью неё можно определить загруженность CPU, загруженность GPU, объем используемой памяти, энергопотребление GPU.

5.Бенчмарк STREAM - это референсный тест, который используется для оценки пиковой пропускной способности памяти.[4]

Финальным шагом в исследовании высокопроизводительных систем является получение результатов запуска на компонентах вычислительных систем.Для оценки производительности необходимо провести серии запусков эталонной задачи на CPU совместно с GPU с использованием средств параллелизма MPI и OpenMP.

В первых сериях запуска необходимо подобрать оптимальное количество MPI процессов и OpenMP потоков при выполнении задачи. Далее определить количество ядер CPU и количество GPU, при котором достигается наибольшая эффективность. После получения подобранных параметров вычислить производительность и эффективность компонент суперкомпьютеров.

Далее рассмотреть полученные данные нормируя на стоимость используемых

компонент, энергопотребление и пиковую производительность. Провести анализ полученных данных.Определить возможные причины приводящие к снижению эффективности использования вычислительных мощностей. Составить рейтинг эффективности архитектур суперкомпьютеров и их компонент по эффективности,производительности, энергопотреблению и стоимости в решении задач.

Список использованных источников

[1] Mark James Abraham и др. “GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers”. В: *SoftwareX* 1 (2015), с. 19—25.

[2] Patrick Carribault, Marc Pérache и Hervé Jourdren. “Enabling Low-Overhead Hybrid MPI/OpenMP Parallelism with MPC.” В: *IWOMP* 6132 (2010), с. 1—14.

[3] Amir Reza Ghods. “A study of Linux Perf and slab allocation sub-systems”. Дис. . . . маг. University of Waterloo, 2016.

[4] Maycon Viana Bordin и др. “DSPBench: A suite of benchmark applications for distributed data stream processing systems”. В: *IEEE Access* 8 (2020), с. 222900—222917.