# ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ НА HADOOP-КЛАСТЕРЕ ДЛЯ ВЫЧИСЛЕНИЯ ЭКСПОНЕНТЫ МАТРИЦЫ

**Паньков А. В., Романчук И. С.**

*УО «Гродненский государственный университет им. Я. Купалы», Гродно, Беларусь, e-mail: irina300694@gmail.com*

Часто при решении систем обыкновенных дифференциальных уравнений с постоянными коэффициентами возникает необходимость нахождения экспоненты матрицы. Возникают проблемы при её вычислении, когда число строк матрицы достигает нескольких миллионов и даже миллиардов. Решение таких систем представляется затруднительным даже для современных вычислительных мощностей.

Одним из вариантов решения данной задачи является реализация алгоритма экспонирования матрицы на вычислительном кластере на базе ПО Hadoop. Hadoop предоставляет пользователю свою распределенную файловую систему Hadoop Distributed File System (HDFS), которая охватывает все узлы кластера. Для реализации параллельного алгоритма был выбран фреймворк Apache Hama – это BSP (Bulk Synchronous Parallel) вычислительный фреймворк над HDFS, который используется для огромных научных вычислений, таких как работа с матрицами, графами и сетями [1].

Для экономии памяти при хранении матриц большого размера на HDFS, Hama предоставляет класс SequenceFile, который хранит данные в байтовом виде как пары ключ–значение. В качестве ключа было решено использовать порядковый номер строки в матрице, а значение – сама строка.

Для вычисления экспоненты матрицы можно воспользоваться разложением экспоненты в ряд Тейлора. Тогда исходная сложная задача сводится к двум более простым задачам: сложение и умножение матриц. Для упрощения выполнения операций умножения было решено один раз транспонировать матрицу, а затем перемножать построчно (скалярное произведение векторов). Чтобы не хранить в памяти каждого узла кластера две матрицы, каждый узел может хранить только часть исходной матрицы и транспонированную. Таким образом, мы добьемся, чтобы все параллельные процессы были абсолютно независимы друг от друга. За счет этого получается выигрыш во времени, т.к. будет минимизировано число дорогостоящих операций, нуждающихся в пересылке данных между узалами кластера.

Возникают две главные проблемы. Как гарантировать отказоустойчивость системы? И как добиться требуемой точности? Решение первой предоставляет сам Hadoop, т.к. HDFS расценивает выход из строя узла данных как норму, а не как исключение. Вторая решается использованием Java класса BigDecimal.

В данный момент основной задачей данного подхода является оптимизация исходного алгоритма для увеличения скорости вычислений, а так же увеличения точности получаемых результатов.

**Литература**