# Оптимизация обработки больших объемов сейсмических данных на примере 3D миграции дуплексных волн

Лавренюк С.И.[1], Лавренюк А.М.[2], Назаренко Е.В.[1]

[1]*Институт кибернетики им. В.М.Глушкова НАН Украины, пр. Глушкова, 40, Киев, Украина*
[2] *Фізико-технічний інститут НТУ України "КПІ"*

lsi@bigmir.net

**Аннотация.** *В работе рассмотрен подход к оптимизации обработки больших объемов данных, на процессорах, состоящих из большого количества ядер. На примере обработки сейсмических данных программой 3D миграции дуплексных волн показано управление самой программой использования вычислительных ресурсов и оперативной памяти для улучшения качества обработки данных и во многих случаях уменьшения времени работы программы.*

## Ключевые слова

кластер, сейсмические данные, геологоразведка, оптимизация.

## 1 Введение

Для качественной и быстрой обработки сейсмических данных в режиме реального времени требуются большие вычислительные ресурсы, которые имеют большие объемы оперативной памяти. При этом эти ресурсы должны быть настроены на выполнение интенсивных файловых операций ввода/вывода. Несмотря на все большую доступность больших вычислительных кластеров, например, как суперкомпьютер ИК НАН Украины (СКИТ) [1], для обработки сейсмических данных, возникают проблемы связанные с ограничениями по объему оперативной памяти и по пропускной способности дисковой системы хранения данных. Программа миграции дуплексных волн [2], [3] состоит из трех основных модулей:

- модуль расчета кубов времен от источников и от приемников, при этом полученные результаты сохраняются в виде файлов;
- модуль выполнения миграции на основе сохраненных файлов времен и сохранение промежуточного результата;
- модуль объединения промежуточного результата всех потоков программы в один результирующий файл.

При работе первого модуля происходит считывание данных из входных файлов, выполняется расчет данных и проводится интенсивная запись в файлы кубов времен и данных промежуточных расчетов. При увеличении числа ядер в одном процессоре и больших объемах входных данных происходит задержка доступа к файловой системе для процессов, в случае если на вычислительном узле кластера запускается k-процессов программы, когда k- число ядер на узле.

Сложность в том, что для работы первого модуля программы необходимо оперативной памяти намного больше, чем для работы двух других модулей. Компенсируется это записью промежуточных результатов расчетов во временные файлы. Чем меньше доступно процессу оперативной памяти, тем чаще происходит обращение к временным файлам. При этом приходится уменьшать качество работы программы иначе она не сможет дать результат за необходимое реальное время. Если же иметь достаточный объем памяти для расчета куба времени без промежуточной записи данных во временные файлы, то программа обработки сейсмических данных будет работать быстрее и самое главное с лучшим качеством.

# 2 Предложенное решение

При возрастании числа ядер на один процессор возникает задержка при работе с операциями записи в файл, когда при запуске задачи на вычислительно узле запускается k-процессов, когда k- число ядер на узле.

На первый взгляд, можно было бы запускать программу, указав при запуске меньшее число ядер, задействованных от одного процессора. Но, тогда при ограниченном количестве доступных ядер получим увеличение общего времени работы программы, так как второму и третьему модулю не нужно много оперативной памяти и намного меньше операций записи на диск, чем в первом модуле.

На первом этапе оптимизации, в программу были внедрены алгоритмы сжатия данных файлов времен без потери качества результата работы программы. Есть два варианта работы программы: упаковка файлов времен при помощи наилучшей Чебышевской аппроксимации сжатия числовых данных [4] (режим 3) и упаковка полученного результата в режиме 3 при помощи алгоритма архивирования LZW (zip) (режим 4) [5]. Например, при обработке одного входного файла на конкретной реальной задаче файлы времен без сжатия занимают 49 Гб. При сжатии только в режиме 3 объем файлов времен уменьшился до 25 Гб. А при работе в режиме 4 объем файлов стал 12 Гб. Таких входных файлов на реальных задачах бывает от 10 до 30, а то и больше.

При этом получено ускорение в работе программы для режима 3 – 1.64 раз, для режима 4- 1.78 раз.

Этот этап оптимизации все же не позволяет уменьшать объем временных файлов, так как здесь качество результата миграции очень чувствительно к качеству промежуточных результатов вычислений, которые при недостаточном количестве оперативной памяти приходится вынужденно сохранять во временных файлах, но без сжатия.

На втором этапе оптимизации предложено в программе выделить виртуальные рабочие группы $Wg_n$, которые состоят из tr ядер, где tr – кратно общему числу ядер в процессоре и подбирается экспериментально, n- число виртуальных рабочих групп, при этом n=k/tr. В каждой такой группе выделяется главное ядро. Запуск программы выполняем точно также: k-процессов, где k- число ядер на узле. Но модуль расчета времен, требующий большого объема оперативной памяти и работающий с файловыми операциями, активно работает только на главном ядре. Остальные ядра из $Wg_n$ задействованы только в вычислительном процессе при помощи нитей, порождаемых процессом на главном ядре. Для создания и управления нитями используется мощный механизм OpenMP [6]. Так как остальные процессы, запущенные не на главных ядрах $Wg_n$ не требуют большого объема памяти, то ее использует процесс главного ядра (формула 1.)

При этом процесс на главном ядре использует под свои нужды объем памяти:

$$RAM^1=RAM^{all}/n, \qquad (1)$$

где $RAM^{all}$ вся оперативная вычислительного память узла, n- к-во виртуальных групп.

На рис. **1** и рис. **2** показано разделение многоядерного CPU $Wg_n$ и объединение ядер в виртуальные рабочие группы.

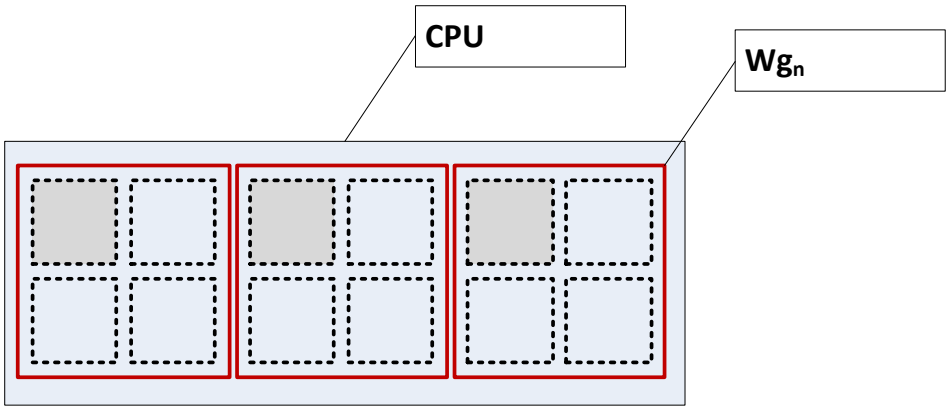


**Рис. 1**. Виртуальные рабочие группы процессора.

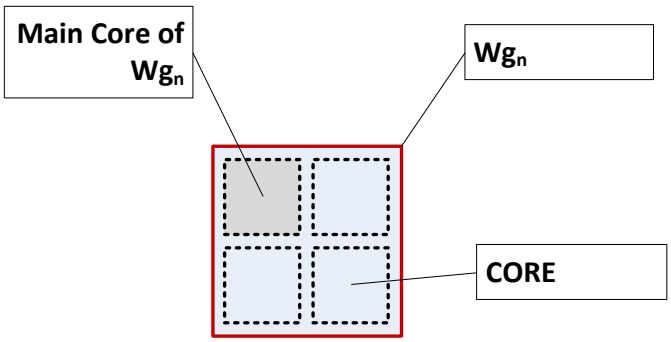


**Рис.2**. Объединение ядер процессора в виртуальную рабочую группу.

После того, как отработал первый модуль, оперативная память перераспределяется (формула 2), так как при работе второго и третьего модуля идет только чтение данных из файлов и не частое записывание результата. При этом каждый процесс работает на своем ядре без порождения нитей.

$$RAM^2=RAM^{all}/k, \quad (2)$$

Соответственно $RAM^2 <= RAM^1$. Так мы получаем дополнительную оперативную память.

Ниже приведен результат полученного ускорения в зависимости от к-ва ядер на процессор и виртуальных групп, в которые ядра объединяются.

**Таблица1**. Результат ускорения при tr=2

| Cores | Tr | Wg | **Told/Tnew** |
|---|---|---|---|
| 4 | 2 | 2 | **1,12** |
| 8 | 2 | 4 | **1,18** |
| 12 | 2 | 6 | **1,22** |

**Таблица2**. Результат ускорения при tr=4

| Cores | Tr | Wg | **Told/Tnew** |
|---|---|---|---|
| 4 | 4 | 1 | **0,91** |
| 8 | 4 | 2 | **1,08** |
| 12 | 4 | 3 | **1,14** |

Следует заметить, что увеличение числа ядер в рабочей группе не дает большого прироста по скорости обработки данных, а иногда может вызвать и замедление. Но при увеличении числа ядер в одной группе можно задействовать больший объем оперативной памяти для главного ядра, а это позволит увеличить качество обработки данных.

# 3 Заключение

В работе показан подход по управлению вычислительными ресурсами и оперативной памятью отдельными модулями одной программы для оптимизации работы программы. Такой подход дает улучшение качества обработки данных и увеличения скорости работы программы.

Предложенный подход позволил увеличить качество обработки сейсмических данных, при ограниченных ресурсах, при этом не только не потеряв в скорости работы программы, а во многих случаях и выигрывая. При этом второй этап оптимизации устраняет те узкие места в работе программы, которые невозможно устранить только сжатием файлов времен.

В дальнейших исследованиях планируется выявление и устранение узких мест в работе третьего модуля, когда происходит в основном интенсивное чтение данных из файлов и не очень интенсивная запись результата. Так же планируется, для интенсивных вычислительных операций использовать графические ускорители (GPU) [7] и язык программирования графических процессоров OpenCl [8], который позволит более универсально работать на устройствах разных производителей [9]. Предварительная оценка времени работы программы [10] показывают положительный результат в выигрыше времени работы программы.

# Литература

[1] Суперкомпьютер ИК НАН Украины – http://icybcluster.org.ua/.

[2] Мармалевський Н.Я. Міграція дуплексних хвиль – новий засіб формування сейсмічних зображень субвертикальних границь / Н.Я.Мармалевський, Ю.В.Роганов, З.В.Горняк та ін. // Зб. наук. праць. УкрДГРІ. – 2007. – № 2.

[3] Marmalyevskyy N. Method, system and apparatus for interpreting seismic data using duplex waves: Patent US / N.Marmalyevskyy, Z.Gornyak, A.Kostyukevych, V.Mershchiy, Y.Roganov. – 2006. – 7. – 110. – 323 B2.

[4] Каленчук-Порханова А.А., Вакал Л.П. Наилучшая Чебышевская аппроксимация для сжатия численной информации // Компьютерная математика. – 2009. – № 1. – С. 99–107.

[5] Назаренко Е.В. Оптимизация обработки больших массивов данных в кластерных системах / Назаренко Е.В., Тульчинский В.Г., Тульчинский П.Г. // Проблеми програмування. – 2010, №2-3 – С. 149-154.

[6] OpenMP – http://openmp.org/.

[7] Li Bo, Liu Guo-feng, Liu Hong. A method of accelerating seismic Pre-stack time migration by GPU . // http://cm.seg.org/documents/10161/997244/1202.pdf

[8] OpenCL – The open standard for parallel programming of heterogeneous systems // http://www.khronos.org/opencl/

[9] А. М. Лавренюк, С. І. Лавренюк Один підхід до вирішення проблеми універсального використання мови програмування OpenCl па різних GPU // Проблеми програмування. – 2012, №2-3 – С. 77-84.

[10] Тульчинский В.Г., Чарута А.К. Оценка времени обработки данных в кластерных системах // Проблеми програмування. – 2006. – №2-3. – С. 118-123.