MPI-IO: Практические аспекты разработки масштабируемых приложений*

Б.Л. Петушкеев
Томский государственный университет

Производительность и доступность вычислительных кластеров в России растет быстрыми темпами и позволяет решать задачи, ранее не доступные. Однако этот качественный переход требует изменения подхода к программированию по причине возрастающих объемов данных. MPI-2 предоставляет средства высокоэффективного параллельного ввода/вывода. В работе рассматривается реализация модели вентилирования в выработанных пространствах шахт с учетом такой возможности.

Пример стандартного подхода (назовем его «обычным») при написании программ, использующих MPI как средство передачи сообщений, предполагает ассоциирование некоторого участка данных в глобальной области решения с определенным процессом. Последний выполняет расчет локального массива, а затем соседние процессы обмениваются «затененными» данными. Особенностью и узким местом «обычного» подхода является выделение памяти для всей глобальной области каждым процессом. С одной стороны, это позволяет упростить написание всей программы в рамках SPMD модели, поскольку каждый процесс использует глобальную индексацию и сшивка всей области не представляет затруднений, с другой – существенно ограничивает возможности исследователя по наращиванию количества задействованных в расчете процессоров (ядер) числом не более нескольких сотен. Типичен пример программы, написанной «обычным» способом и запущенной на, например, кластере СКИФ Cyberia МВЦ ТГУ, но достигающей насыщения уже при 250–300 процессах и показывающей практически останов в росте производительности из-за чрезмерно малой гранулярности расчетов. Следует отметить, что на данной машине на 1 ядро приходится всего 1GB оперативной памяти, что при обычной практике запуска MPI-процесса на каждом ядре ведет в итоге к существенному ограничению размерности решаемой задачи.

* Работа выполнена при финансовой поддержке РФФИ № 08-08-12029-офи.

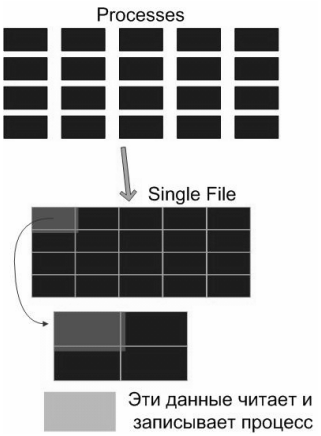


Рис. 1. Обмен данными

Для обхода подобных узких мест в работе предлагается избавиться от глобального массива, необходимого в «обычном» подходе для сборки решения в целях изучения каких-либо интегральных характеристик и, наконец, вывода результатов расчета в файл. Вместо этого предполагается, что на всем протяжении расчета никто из процессов не «видит» полностью всю картину решения, а вывод производится напрямую каждым процессом (рис. 1). При этом если производить запись каждым процессом в свой уникальный файл, то очень скоро встанет вопрос управления и обслуживания огромного количества, что само по себе представляет большую отдельную задачу.

Удачным примером может служить программа FLASH, генерирующая при запуске на 32000 узлах суперкомпьютера BG/L более 72 млн файлов. Вместе с тем возникнут неизбежные проблемы при попытке рестарта программы с контрольной точки на отличном от изначального количестве процессоров.

К параллельной стратегии оперирования с файлами приводят следующие основные доводы:

- последовательный I/O в больших задачах становится узким местом;
- для работы с глобальным файлом (слияние, извлечение и т.д.) требуется трудоемкий пост- и препроцессинг.

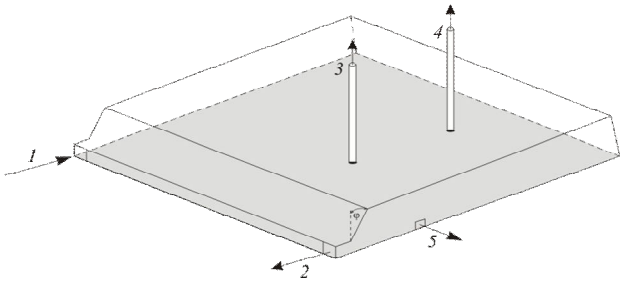


Рис. 2. Область исследования

Предложенный подход к реализации параллельных программ был применен к задаче моделирования процесса фильтрации метано-воздушной смеси через пористую среду шахтных выработок.

Пример физической области расчетной задачи показан на рис. 2, где введены обозначения: *1* – сечение конвейерного штрека; *2* – сечение вентиляционного штрека; *3* и *4* – газоотсасывающие скважины; *5* – открытая сбойка на газодренажный штрек.

Особенностью рассматриваемой проблемы являются чрезвычайно низкие скорости движения газа, составляющие порядка нескольких см/с . С учетом размеров выработанных пространств действующих шахт задача не представляется разрешимой в разумные сроки на обычном ПК. К тому же вкупе с применением суперкомпьютера применяется процедура ускорения, предложенная в [1], что позволяет получать итоговые результаты в очень короткое время.

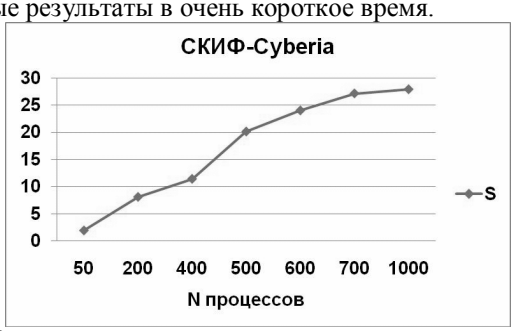


Рис. 3. Зависимость ускорения от числа используемых процессоров

На рис. 3 представлены результаты экспериментов на ускорение итоговой программы при числе ячеек около 24 млн. Объем требуемой оперативной памяти превышает 18 Гб, поэтому величины ускорения отнесены ко времени расчета на 25 процессорах (ядрах). Отметим

практически линейный характер кривой ускорения с наметившейся просадкой в районе 1000 задействованных ядер. Это закономерный результат, поскольку такое количество точек уже недостаточно для загрузки процессоров и на показатели сильно влияют накладные расходы в виде задержек при передаче сообщений.

Также проведены эксперименты на операциях чтения/записи трехмерного массива вещественных чисел double в/из файла общим размером 4Гб при различных режимах работы MPI-IO. Массив на сетку процессов распределялся блочно в форме (block, block, block), часто используемой в научных приложениях. Целевая платформа кластера СКИФ-Cyberia использует параллельное хранилище ReadyStorage ActiveScale Cluster, основанное на файловой системе PanFS, с заявленной производительностью на уровне 700 Мб/с.

На рис. 4 представлены результаты производительности файловой системы на операциях чтения/записи в зависимости как от уровня вызовов MPI-IO, так и от количества задействованных процессов. Пропускная способность рассчитывалась как объем переданных данных, отнесенный к максимальному времени чтения/записи любого из процессов. На обеих диаграммах не приведены результаты, соответствующие level-0 и level-1 вызовам MPI-IO, показавшие одинаково плохие ре-

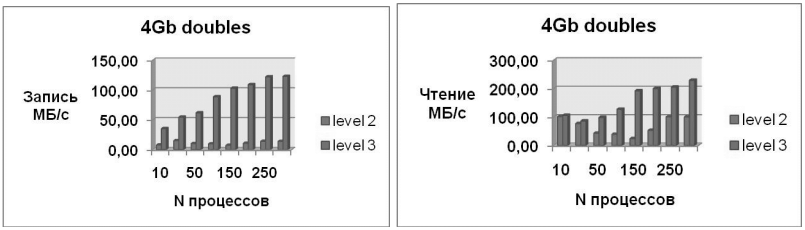


Рис. 4. Производительность файловой системы

зультаты при выбранной политике доступа к файлу. Level-3 демонстрирует стабильный рост показателей при увеличении количества процессов, что указывает на хорошую масштабируемость работы файлового хранилища. Хорошо заметна зона прогиба на операции чтения в районе 10–100 процессов, причем такая же картина наблюдается и при меньшем объеме файла (1 Гб). Объяснение этому, полагаем, следует искать в работе механизма data-sieving, точнее в условиях, при которых он себя эффективно проявляет. Level-2 показывает низкие результаты при записи, связанные с блокировкой файла, и, как следствие, постоянные конфликты за ресурс. Интересно ведут себя level-2 запросы при чтении файла: падение до очень низких показателей, а затем быстрый рост. Level-2 генерирует дополнительные обращения к дис-

кам из-за отсутствия координации, а также наличия больших расстояний между нужными данными (о поиске компромисса в работе data sieving пишут сами авторы ROMIO). При увеличении же количества процессов сверх 200 блоки данных в файле становятся небольшими и эффективность механизма data sieving улучшается, возможно, из-за попадания в буфер ROMIO (по умолчанию 4 Мб) сразу нескольких нужных блоков.

Предлагаемый подход показал свою применимость и актуальность в свете появления в России кластера, достигшего полупетафлопной производительности. Показатели производительности параллельной ФС ожидались лучше, но все же они гораздо выше последовательного I/O (следует иметь в виду, что файл в итоге единый!).

Литература

1. **Васенин И.М., Петушкеев Б.Л.** Задача об ускорении решения совместных уравнений газовой динамики и переноса // Известия вузов. Физика. 2007. №9/2. С. 274-281.