ПРОЗРАЧНАЯ АКСЕЛЕРАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

А.А. Кошулько

Чем шире становится круг пользователей высокопродуктивных вычислительных систем, тем большее значение приобретает дружественность прикладных программ используемых в таких системах. Отказ от командной строки и переход к графическим интерфейсам значительно повысил бы продуктивность и доступность использования высокопродуктивных вычислений.

Однако, современная концепция многопользовательских высокопродуктивных систем с распределенной памятью не предусматривает прямой возможности предоставления графических интерфейсов. Связанные с этим проблемы достаточно показательны в случае программного обеспечения для интеллектуального анализа данных. Популярные визуальные программные пакеты, такие как Rapid Miner, KNIME, SPSS Modeler, SAS Enterprise Miner, не поддерживают MPI-кластеры. В то же время, известно, что существуют MPI-реализации для наиболее распространенных методов машинного обучения, используемых в перечисленных программных системах, в частности, это нейронные сети [1], деревья решений [2], метод опорных векторов [3], метод k ближайших соседей [4] и метод k-средних [5].

В отличии от MPI-систем, поддержка многопроцессорных компьютеров с общей памятью выполнена в популярных пакетах на достаточном уровне. Более того, не является редкостью реализация распределенных вычислений (KNIME, SAS). При этом узлы распределенной системы создаются из обычных копий одной и той же программы, запущенной на разных компьютерах. Таким образом, общей особенностью распределенных систем и компьютеров с общей памятью является возможность запускать программы в графическом режиме.

Можно сделать вывод, что для решения проблем использования высокопродуктивных MPI-систем в интеллектуальном анализе данных, достаточно обеспечить прозрачное (незаметное) подключение MPI-системы к персональному компьютеру пользователя, то есть, освободить пользователя от низкоуровневой работы с удаленной вычислительной системой. Это, в свою очередь, потребует прозрачного перемещения пользовательских данных на всех стадиях вычислений, от предобработки, до визуализации, и, кроме того, прозрачного управления состояниями удаленных вычислительных процессов. Поскольку речь идет о прозрачном, с точки зрения пользователя, использовании высокопродуктивных вычислений, то удаленную систему можно считать акселератором персонального компьютера (ПК). Заметим, что данная концепция, в принципе, также позволяет организовать гибридные вычисления.

Примером осуществления описанной концепции является программная система GMDH Shell [6] для операционной системы Windows. Система предоставляет два параллельных алгоритма интеллектуального анализа данных – самоорганизующиеся нейронные сети и комбинаторную структурную оптимизацию моделей. Кроме многопоточного локального счета, система способна к прозрачной акселерации вычислений с помощью Linux-кластера.

Среда GMDH Shell используется для решения задач интеллектуального анализа данных, таких как классификация, прогнозирование, анализ временных рядов. Вычислительный процесс в данной программной системе основан на методе группового учета аргументов (МГУА) [7]. С помощью МГУА происходит структурно-параметрическая оптимизация математических моделей, отображающих закономерности много-параметрических данных исследуемого объекта.

Определение структуры модели и её параметров представимо в виде итерационного процесса, который распараллеливается для достижения большей производительности. Если объем данных анализируемого объекта позволяет их передачу от ПК пользователя к удаленному акселератору за приемлемое время, то предложенная схема использования высокопродуктивных вычислений будет эффективна.

Программная система состоит из двух частей – “локальной” и “удаленной”. Локальная программа устанавливается на ПК пользователя, и будучи полностью самодостаточной, может подключать Linux-кластер в качестве акселератора вычислений. Взаимодействие с Linux-кластером происходит с использованием протоколов SSH, SCP или SFTP. Такой подход позволил использовать удаленную вычислительную систему без предварительной инсталляции прикладного программного обеспечения, поддерживать работу с различными управляющими оболочками вычислительных систем, выполнять резервирование вычислительных ресурсов для работы в режиме сессии.

Локальная часть программы должна обладать информацией о сетевом адресе удаленной системы, имени и пароле зарегистрированного в ней пользователя, именах команд системы управлениями задачами.

Алгоритм взаимодействия с удаленной системой будет иметь следующий вид.

1. Соединение с удаленной системой по протоколу SSH и создание рабочего каталога, в том случае, если он не был создан ранее.

2. Если в рабочем каталоге отсутствуют исполняемые файлы удаленной части программы, то они копируются по протоколу SCP или SFTP и получают статус исполняемых файлов, а если присутствуют, то производится проверка, не запущены ли они.
3. Если работающие копии серверной программы не обнаружены, то производится копирование данных предназначенных для удаленной обработки на кластере и запуск удаленной программы посредством команд системы управления заданиями;
4. Если обнаружена уже запущенная серверная часть программы, то спрашивается разрешение на передачу нового вычислительного задания.
5. Далее, локальная программа проверяет готовность результатов удаленных вычислений с заданным интервалом.
6. Если обнаружены завершенные результаты вычислений, они копируются на ПК пользователя для дальнейшего их использования локальной программой.

Для ведения переговоров между параллельно исполняемыми частями программы, используются файлы-индикаторы, говорящие об этапах исполнения задания на удаленной системе.

- Индикатор A – создается удаленной программой сразу после ее запуска, а убирается непосредственно перед закрытием программы.
- Индикатор B – создается программой-сервером на время выполнения очередного задания.
- Индикатор C – опционально создается локальной программой в рабочем каталоге удаленной программы, что сигнализирует о необходимости перехода в режим ожидания новых заданий после исполнения текущего задания. Длительность ожидания регулируется отдельным параметром и продлевается на заданный интервал в случае обнаружения признаков активности со стороны локальной программы.

Файлы-индикаторы обеспечивают успешное взаимодействие между двумя частями программы несмотря на обрывы связи, то есть, не требуют постоянного соединения локальной программы с удаленным вычислителем.

Необходимо отметить, что работы над дружественными интерфейсами прикладных программ для высокопродуктивных вычислений развиваются в нескольких направлениях, в частности, авторы работы [8] идут по пути предоставления веб-интерфейсов. Такой подход является менее прозрачным с точки зрения пользователя, но имеет неоспоримые преимущества, в том случае, когда объем обрабатываемых данных не позволяет копировать их с удаленной системы на компьютер пользователя.

Реализованная в программной системе GMDH Shell концепция прозрачной акселерации персонального компьютера с помощью удаленной высокопродуктивной системы, обладает следующими преимуществами:

- предоставляет одинаковый интерфейс прикладной программы для работы в локальной и удаленной системе;
- предусматривает резервирование акселератора для интерактивной работы в режиме сессии.

Недостатками концепции можно считать проблемы, которые могут возникнуть при обработке больших объемов данных. Тогда потребуется проводить дальнейшие усовершенствования, например, за счет добавления возможностей удаленного управления хранилищами данных.

Последующее развитие программной системы GMDH Shell будет направлено на добавление новых алгоритмов интеллектуального анализа данных и добавление поддержки прозрачного использования локальных и удаленных акселераторов на основе GPU.

ЛИТЕРАТУРА:

1. Udo Seiffert. Artificial Neural Networks on Massively Parallel Computer Hardware. // ESANN'2002 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 319-330.
2. O. T. Yıldız, O. Dikmen. Parallel univariate decision trees. // Pattern Recogn. Lett. 28, 7 (May. 2007), 825-832.
3. Kristian Woodsend, Jacek Gondzio, Hybrid MPI/OpenMP Parallel Linear Support Vector Machine Training. // Journal of Machine Learning Research 10 (2009) pp. 1937-1953.
4. G. Apar´ıcio, I. Blanquer, V. Hern´andez, A parallel implementation of the k nearest neighbours classifier in three levels: Threads, mpi processes and the grid. // Proceedings of the 7th Meeting of the International Meeting on High Performance Computing for Computational Science, 2006, Porto, Portugal.
5. M.N. Joshi, Parallel K-Means Algorithm on Distributed Memory Multiprocessors. // Computer, 2003
6. O.A. Koshulko, A.I. Koshulko. Acceleration of GMDH combinatorial search with HPC clusters // Proceedings of 2nd International Conference on Inductive Modelling , ICIM2008, ISBN 978-966-02-4889-2, September 15-19, Kyiv – 2008, pp.164-167.
7. А.Г. Ивахненко, В.С. Степашко, Помехоустойчивость моделирования. / Киев: Наукова думка, 1985. – 214 с.
8. В.А. Вознесенский, В. Неверов, К популяризации научных приложений на инфраструктурах EGEE и ГридННС // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: Труды Всероссийской суперкомпьютерной конференции, ISBN 978-5-211-05697-8 (21-26 сентября 2009 г., г. Новороссийск). - М.: Изд-во МГУ, 2009. - с. 468-470.