ным (ожидающим изменения сигналов). Следующее состояние модели определяется состояниями составляющих ее процессов, состояниями формирователей сигналов и состоянием списка процессов, заблокированных на время. Вычисляется после обработки ближайшего события (изменения сигналов в формирователях, разблокирования процессов), состоящей в попытке последовательного запуска каждого из активных процессов, начиная с текущего состояния.

Таким образом, результатом выполнения п.п. 1-4 для произвольного модельного описания служит однородная промежуточная модель в терминах языка VHDL, представляющая собой частный случай процессного описания проекта. А графовое представление каждого параллельного оператора process, состоящего из комбинации последовательных операторов, может служить основой для реализации моделирования. При этом процесс моделирования проекта интерпретируется как пошаговое изменение его состояний в соответствии с графами процессов и результатами вычислений в их предикатных узлах. Рассмотрены эквивалентные формы параллельных операторов, графы и варианты реализации процессов, пригодные для генерации исполнимых кодов и организации моделирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бибило П.Н. Основы языка VHDL. - М.: “СОЛОН-Р”, 2000. – 210 с.
2. Сергиенко А.М. VHDL для проектирования вычислительных устройств. - К.: “Корнейчук”, 2003. – 208 с.
3. Армстронг Дж. Р. Моделирование цифровых систем на языке VHDL. - М.: Мир, 1992. – 175 с.
4. Поляков А.К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. - М.: “СОЛОН-Пресс”, 2003. – 320 с.
5. Прихожий А.А., Муравьев Г.Л. Система проектирования цифровых СБИС с языком VHDL на ПП ЭВМ. Разработка и использование ПЭВМ// Труды международного симпозиума INFO' 89. - Мн., 1989. - т.2, ч.1. – 6 с.
6. Максимей И.В. Имитационное моделирование на ЭВМ. - М.: Радио и связь, 1988. – 232 с.
7. Прихожий А.А., Муравьев Г.Л. и др. Система автоматизированного проектирования СБИС. Процедуры проектирования. Материалы по математическому обеспечению ЭВМ. - Мн., ИТК АН РБ, 1992. – 98 с.
8. Дудкин А.А., Головко В.А. , Муравьев Г.Л. и др. Алгоритмы и подсистемы автоматизированного логического проектирования цифровых СБИС. Материалы по математическому обеспечению ЭВМ. - Мн., ИТК АН РБ, 1994. – 120 с.
9. Муравьев Г.Л., Мухов С.В. Подход к процессной интерпретации VHDL-моделей// Сборник материалов VII международной научно-методической конференции “Наука и образование в условиях социально-экономической трансформации общества”. – Брест: ИСЗ, 2004. - Ч.2.

УДК 681.324

Савицкий Ю.В.

ПАРАЛЛЕЛЬНАЯ СИСТЕМА ОБРАБОТКИ ХАОТИЧЕСКИХ ПРОЦЕССОВ НА БАЗЕ ПРОТОКОЛА MPI

Развитие фундаментальных и прикладных наук, технологий требует применения все более мощных и эффективных методов и средств обработки информации. В качестве примера можно привести разработку реалистических математических моделей, которые часто оказываются настолько сложными, что не допускают точного аналитического их исследования. Единственная возможность исследования таких моделей, их верификации (то есть подтверждения правильности) и использования для прогноза - компьютерное моделирование, применение методов численного анализа. Другая важная проблема - обработка больших объемов информации в режиме реального времени. Все эти проблемы могут быть решены лишь на достаточно мощной аппаратной базе, с применением эффективных методов программирования [1].

Переход на качественно новый уровень производительности потребовал от разработчиков ЭВМ системных решений, значительная часть которых основана на организации параллельной обработки. Параллелизм все более широко используется как средство, позволяющее повышать производительность вычислительных систем, не прибегая к повышению быстродействия их компонентов. *Основная цель состоит в выборе такого отображения задачи в реализующую ее техническую структуру, при котором параллелизм мог бы использоваться для уменьшения времени решения задачи, не вызывая чрезмерного роста числа избыточных обрабатывающих модулей и стоимости передачи данных* [*1,2*].

Стиль программирования, основанный на параллелизме задач подразумевает, что вычислительная задача разбивается на несколько относительно самостоятельных подзадач и каждый процессор загружается своей собственной подзадачей. Компьютер при этом представляет собой MIMD - машину. Аббревиатура MIMD обозначает в известной классификации архитектур ЭВМ вычислительную систему, выполняющую одновременно множество различных операций над множеством, вообще говоря, различных и разнотипных данных. Для каждой подзадачи пишется своя собственная программа. Чем больше подзадач, тем большее число процессоров (вычислительных модулей, ВМ) можно использовать, тем большей эффективности можно добиться. Важно то, что все эти программы должны обмениваться результатами своей работы. Практически такой обмен осуществляется вызовом процедур специализированной библиотеки. Программист при этом может контролировать распределение данных между процессорами и подзадачами и обмен данными. Очевидно, что в этом случае требуется определенная работа для того, чтобы обеспечить эффективное совместное выполнение различных программ.

Примерами специализированных библиотек являются библиотеки *MPI (Message Passing Interface)* и *PVM (Parallel Virtual Machines)*. Эти библиотеки являются свободно распространяемыми и существуют в исходных кодах.

MPI – это разработанная Аргоннской Национальной Лабораторией (США) большая библиотека функций для передачи сообщений, которая позволяет преобразовать последовательную программу в программу, которая будет *полностью использовать параллельную архитектуру используемой вычислительной системы*. MPI предоставляет программисту единый механизм взаимодействия ветвей внутри параллельного приложения независимо от машинной архитектуры (однопроцессорные / многопроцессорные с общей памятью / раздельной памятью), взаимного расположения ветвей (на одном процессоре (ВМ) / на разных процессорах (ВМ)).

Для MPI необходимо создавать приложения, содержащие код всех вервей сразу. MPI-загрузчиком запускается указываемое количество экземпляров программы. Каждый экземпляр определяет свой порядковый номер в запущенном коллективе, и в зависимости от этого номера и размера коллектива выполняет

Рис. 1. Последовательность этапов обучения нейронных сетей в параллельном режиме.

ту или иную ветвь алгоритма. Такая модель параллелизма называется Single Program/Multiple Data (SPMD) и является частным случаем модели Multiple Instruction/Multiple Data (MIMD). Каждая ветвь имеет пространство данных, полностью изолированное от других ветвей. Обмениваются данными ветви только в виде сообщений MPI. Все ветви запускаются загрузчиком одновременно. Количество ветвей фиксировано – это означает, что в ходе работы порождение новых вервей невозможно.

Разные MPI-процессы могут выполняться как на разных процессорах (ВМ), так и на одном и том же – для MPI-программы это роли не играет, поскольку в обоих случаях механизм обмена данными одинаков. Процессы обмениваются друг с другом данными в виде сообщений. Сообщения проходят под идентификаторами, которые позволяют программе и библиотеке связи отличать их друг от друга. Для совместного проведения тех или иных расчетов процессы внутри приложения объединяются в группы. Каждый процесс может узнать у библиотеки связи свой номер внутри группы, и, в зависимости от номера, приступает к выполнению соответствующей части расчетов. MPI-ветвь запускается и работает как обычный процесс, связанный через MPI с остальными процессами, входящими в приложение. В остальном процессы следует считать изолированными друг от друга – у них разные области кода, стека и данных.

Описанные выше свойства и возможности MPI были применены автором для разработки параллельного MPI-приложения в рамках нейросетевой системы анализа хаотических процессов.

В процессе многочисленных экспериментов с программными компонентами, реализующими различного рода нейросетевые алгоритмы, установлено, что наиболее сложными в вычислительном и временном аспекте являются алгоритмы обучения многослойных нейронных сетей, основанные на методе обратного распространения ошибки. Установлено также, что вычислительная сложность обучающих процедур зависит от: размера обучающего множества; количества итераций обучения; сложности архитектуры нейронной сети (количества и типа нейронов, наличия рекуррентных связей); типа используемого алгоритма обучения.

Специфика методов и алгоритмов обработки хаотических сигналов состоит в том, что при решении исследовательских и прикладных задач (например, анализ структуры данных ЭЭГ и ЭКГ, прогнозирование сложных временных процессов) требуется работа одновременно со значительным числом наборов данных, каждый из которых содержит большое количество наблюдений (тысячи и десятки тысяч измерений). В результате реализация стадии нейросетевого обучения в алгоритмах обработки характеризуется неприемлемо большими временными затратами в случае использования одного ВМ.

Разработанное ПО представляет собой консольное приложение, реализующее процедуры обучения нейронных сетей в параллельном режиме на базе набора ВМ, объединенных общей коммуникационной средой. Обученные нейронные сети используются далее в данной программной системе для осуществления нейросетевого анализа хаотических сигналов [3]. Последовательность этапов параллельного обучения приведена на рис. 1.

При запуске режима обучения выполняется инициализация MPI-окружения и создание прикладных процессов в соответствии с конфигурацией MPI-приложения, размещаемой в файле конфигурации **NnSim.pg**. В данном файле указывается количества процессов и сетевые имена ВМ, в которых эти процессы должны быть инициализированы. После этого каждый процесс путем вызова специальной MPI-функции определяет собственный идентификационный номер (ранг), который затем используется для спецификации работы (подзадачи, которую должен выполнить данный процесс).

Следующим этапом является чтение каждым процессом файла спецификации задач **NnTask.t** и определение на основании своего ранга своей подзадачи. Структура файла **NnTask.t** приведена на рис. 2. Файлы проекта, имеющие стандартные имена NNETXX.DAT (XX – номер процесса, закрепленного за данным проектом), содержат всю информацию, необходимую для создания нейросетевой модели: конфигурацию нейронной сети, начальные значения весовых коэффициентов нейронов, эталоны обучающей выборки и некоторые другие параметры.

После загрузки файлов проектов процессы осуществляют обучение в соответствии с режимом обучения, указанным в спецификации. Отличительной особенностью параллельного приложения является возможность задания режима и параметров обучения индивидуально для каждого процесса, что значительно увеличивает гибкость при решении различного рода прикладных и исследовательских задач. Информирование об исключительных ситуациях, возникающих при работе процессов (ошибки выполнения, завершение обучения), осуществляется в виде MPI-сообщений, передаваемых в мастер-процесс. В качестве мастер-процесса выступает процесс, имеющий нулевой ранг

```
MPI TASK SPECIFICATION
BEGIN
<номер (ранг) процесса [0]>:
<имя файла проекта [NNET00.DAT]>;
<параметры обучения>.
<номер (ранг) процесса [1]>:
<имя файла проекта [NNET01.DAT]>;
<параметры обучения>.
. . .
<номер (ранг) процесса XX>:
<имя файла проекта [NNETXX.DAT]>;
<параметры обучения>.
. . .
END
```

Рис. 2. Структура файла спецификации задач.

В ходе проведения вычислительных экспериментов по анализу данных ЭЭГ различной размерности была исследована эффективность MPI-приложения по критерию временной сложности. В качестве аппаратных средств было использовано 5 ВМ (ПЭВМ класса Pentium II, III) в составе локальной вычислительной сети Fast Ethernet с пропускной способностью канала 100 Мбит/с. В таблице 1 приведены усредненные результаты для 10 попыток обучения многослойных нейронных сетей с 7 входными, 5 скрытыми сидмодальными и 1 выходным линейным нейронами. Были использованы 5 различных наборов данных ЭЭГ. В качестве сравнительного критерия использован коэффициент изменения времени обработки за счет MPI-параллелизации, определяемый как:

$$K = \frac{t_{MPI}}{t_1},$$

где t_1 - время выполнения обучения всех наборов данных на одном ВМ;

t_{MPI} - время выполнения обучения всех наборов данных на пяти ВМ в режиме MPI.

Указанные выше параметры представляют собой интервалы времени между моментом запуска приложения на выполнение до момента завершения работы всех процессов.

Таблица 1. Результаты тестирования MPI-приложения в задаче обучения нейронных сетей

Размер обучающей выборки	**Количество итераций обучения**	**K**
2048	1000	0.297
2048	2000	0.293
2048	3000	0.291
4096	1000	0.285
4096	2000	0.278
4096	3000	0.274

Из анализа приведенных данных следует: эффективность MPI-приложения несколько уменьшается за счет временных затрат на выполнение служебных функций (инициализации MPI-процессов, пересылок между процессами и др.); эффективность MPI-приложения может увеличиваться при увеличении размера обучающей выборки и количества выполненных итераций алгоритма. Последние факторы обусловлены увеличением удельной доли вычислений, связанных непосредственно с обучением нейронной сети в параллельном режиме.

Таким образом, разработанная параллельная MPI-система позволяет одновременно осуществлять обучение и формирование нейросетевых моделей на множестве вычислительных модулей, входящих в состав многокомпонентной вычислительной системы. Система обеспечивают гибкое конфигурирование параллельного приложения, что позволяет настраивать систему на параллельную работу с различным числом обучающих множеств, осуществлять различные режимы обучения и тем самым значительно расширить область практических задач, в которых могут быть эффективно применены нейросетевые методы.

Автор выражает благодарность Белорусскому республиканскому фонду фундаментальных исследований Республики Беларусь за оказанную поддержку при проведении данных научно-исследовательских работ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Корнеев В.В. Параллельные вычислительные системы. – М.: «Нолидж», 1999. – 320с.
2. В.В. Воеводин, Вл.В. Воеводин. Параллельные вычисления. – СПб.: «BHV-Петербург», 2002. – 609с.
3. V. Golovko, Y. Savitsky, N. Maniakov. Neural Networks for Signal Processing in Measurement Analysis and Industrial Applications: the Case of Chaotic Signal Processing // chapter of NATO book “Neural networks for instrumentation, measurement and related industrial applications”. - Amsterdam: IOS Press, 2003, pp. 119-143.