Являющиеся детьми данного объекта. Возможны и другие варианты поиска зависимых объектов.

Список использованных источников

1. Советов, Б.Я. Базы данных: теория и практика: учебник для студентов вузов, обучающихся по направлениям; Информатика и вычислительная техника; / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовский. - 2-е изд. - Москва: Юрайт, 2012. - 462, [1] с.
2. Спецификация и форматы обмена данными в разнородных информационных системах на базе XML-технологий [Электронный ресурс]. - Режим доступа: http://citforum.ru/internet/xml/xmltech (дата обращения: 5.10.2016).
3. Интуит – национальный открытый университет. – Электрон. Дан. Режим доступа: http://www.intuit.ru.
4. Мурат, Йенер, Алекс Фидом. Java EE. Паттерны проектирования для профессионалов - Питер, 2016. – 240 с.

УДК 004

Б.А. Кирьяк

ОПТИМИЗАЦИЯ БЫСТРОДЕЙСТВИЯ ПРОГРАММ НА ОСНОВЕ СЕТЕЙ ПЕТРИ ПРИ ПОМОЩИ РАСПАРАЛЛЕЛИВАНИЯ С ИСПОЛЬЗОВАНИЕМ ИНТЕРФЕЙСА MPI

Научный руководитель: к.ф.-м.н., доцент Ю.Ю. Горюнов
Пензенский государственный университет
Boriskiryak@yandex.ru

Цель данной работы – улучшение быстродействия работы программы, моделирующей работу сетей Петри. При большом количестве переходов, последовательный поиск в сети занимает довольно значительный объем времени. Для оптимизации работы широкое распространение получили методы распараллеливания данных, которые позволяют значительно ускорить время обработки массива переходов в сетях Петри.

Сети Петри – средство моделирования динамических систем. Анализ сетей Петри помогает получить важные сведения о структуре и поведении моделируемых систем. Полученные сведения используются для оценки систем и выработки предложений по их усовершенствованию.

Сеть Петри состоит из двух компонент: базовой сети и маркировки. Базовая сеть представляет собой двудольный мульти орграф, вершины одной доли принято называть позициями и обозначать кружками, а вершины второй доли называют переходами и обозначают вертикальными отрезками.

Маркировкой сети называется кортеж весов позиций. Переход t_j называется активным при маркировке (m_1, …, m_n), если для всех позиций p_i

выполняется неравенство $m_i \geq I(p_i, t_j)$, где $I(p_i, t_j)$ означает множество всех дуг из позиции p_i в переход t_j (множество всех входов перехода t_j).

Если переход t_j является активным, то он *срабатывает*, что переводит маркировку $(m_1, \ldots, m_n)$ в новую маркировку $(M'_1, \ldots, M'_N)$, значения весов позиций в которой определяется равенством

$$M'_I = M_I + O(T_J, P_I) - I(P_I, T_J),$$

В котором $O(T_J, P_I)$ Означает множество всех дуг из перехода t_j в позицию p_i (множество всех выходов перехода t_j).

Маркировка M_2 называется достижимой из маркировки M_1, если существует последовательность активных переходов, срабатывание которых переводит маркировку M_1 в M_2. Множество всех достижимых маркировок из начальной маркировки называется множеством достижимости сети Петри.

Очевидно, что отношение достижимости ρ на множестве достижимости маркировок D является не рефлексивным и транзитивным, поэтому (D, ρ) является орграфом, который называют орграфом достижимости сети Петри.

На рис. 1 приведён пример сети Петри с начальной маркировкой (3, 0, 0, 1) и её орграфа достижимости.

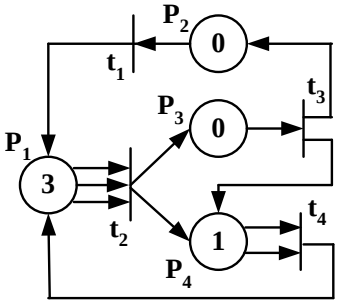
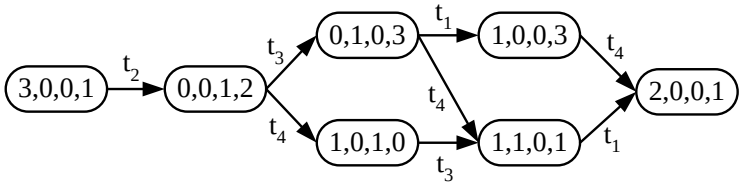
Сеть Петри	Орграф достижимости
	

Рис. 1. Пример сети Петри и орграф достижимости

Орграф достижимости сети Петри позволяет получать знания о динамической системе средствами теории графов.

В силу того, что зачастую для серьезных проектов строятся сети Петри с весьма значимым количеством переходов, быстродействие в таких сетях падает. Поэтому оптимизация скорости работы систем построенных с помощью сетей Петри является важной задачей. Вопрос реализации сетей Петри на императивных языках программирования освящен крайне слабо, а темы многопоточной реализации таких программ касаются еще реже. Поэтому данная работа будет весьма актуальна в сегодняшнее время, когда компьютерные технологии позволяют разделять задачу и решать ее одновременно, получаю большой прирост в скорости выполнения.

Для многопоточной реализации было решено использовать средства распараллеливания данных библиотеки языка C++ – MPI (Message Passing Interface) [2]. Этот программный интерфейс позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. MPI является

наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров.

Существенным плюсом многопоточной реализации на основе MPI по сравнению с реализациями на основе интерфейсов Windows API и pthreads является возможность переноса кода на различные платформы, низкое время ожидания между пересылкой данных и синхронизацией, возможность запуска программы последовательно на одноядерных машинах, и несущественное изменение кода программы. Также MPI является высокоуровневым интерфейсом, что позволяет программисту экономить время, не прописывая многие строки ненужного кода вручную.

Для реализации последовательной программы, моделирующей работу сети Петри, был использован класс и функции, описанные в [3]. Впоследствии в программу была добавлена многопоточная реализация на основе интерфейса MPI.

В автоматическом режиме программа случайным образом выбирает переход, если он доступен, переход выполняется, программа ищет разрешенные переходы. Это повторяется, пока не будет выполнено указанное число переходов. При выполнении программы в автоматическом режиме происходит подсчёт времени работы каждого процесса, для сравнения их скорости работы. Так же ведётся подсчёт времени выполнения всей программы – оно выводится в главном процессе(ROOT).

Распараллеливание заключается в разделении массива, содержащего условия выполнения переходов, на равные сегменты для каждого процесса. Главный процесс (ROOT) рассчитывает размер сегмента и с помощью функции MPI_Scatter рассылает каждому процессу, в том числе и себе, границы выделенного ему сегмента в массиве. Получив границы массива, каждый процесс выполняет поиск разрешенных переходов в рамках своего сегмента, результаты поиска записываются во внутренний буфер.

Завершив поиск в своём сегменте, каждый процесс отправляет свои результаты всем другим процессам. Для этого используется функция MPI_Alltoall, она заменяет две функции, применяющиеся для сбора частей данных с результатами поиска каждого из процессов в главном процессе (MPI_Gather) и рассылке полных данных между процессами (MPI_Scatter). Полные данные требуются для последующих расчётов состояния.

В итоге, при параллельном выполнении каждый процесс будет перебирать сегмент массива, равный (P*S)/N, а при последовательном P*S, где P – число переходов, S – число состояний, N – число процессов. Сравнение результатов работы программ, работающих последовательно и параллельно представлено в табл. 1.

Таблица 1

Результаты работы

№ Запуска	Время выполнения	
	Последовательный	MPI

1	7.892	4.820
2	8.116	4.797
3	7.944	4.854
4	7.912	4.834
5	8.064	4.825
Среднее	7.986	4.826

Для выполнения было задано 10000 переходов. Программа запускалась на машине с двумя физическими ядрами, но четырьмя логическими процессорами. Программы, выполняющиеся в параллельном режиме с помощью MPI, работают быстрее на 3.16 секунд (39.6%), чем программа, работающая в последовательном режиме. Такие результаты можно получить при большом объёме данных – при малом объёме разница во времени будет незначительна.

Как видно из результатов, метод многопоточной реализации успешно применим к программам работающим на основе сетей Петри, и позволяет значительно оптимизировать время выполнения программы. В дальнейшем данный метод распараллеливания также планируется применить к программам обработки ориентированных графов достижимости сетей Петри.

Список использованных источников

1. Котов, В. Е. Сети Петри. – М.: Наука, 1984. – 160 с.
2. Оленев Н.Н. Основы параллельного программирования в системе MPI. М.: ВЦ РАН. 2005. – 80 с.
3. Федотов И.Е. Модели параллельного программирования. – М.: СОЛОН-ПРЕСС, 2012. – 384 с.

УДК 004.77

Д.В. Кирюхин

ИССЛЕДОВАНИЕ МЕТОДОВ ТЕХНОЛОГИИ ВИРТУАЛИЗАЦИИ РАБОЧИХ СТОЛОВ В ОБРАЗОВАТЕЛЬНОМ УЧРЕЖДЕНИИ

Научный руководитель: О.М. Гущина, канд. Пед. Наук, доцент
Тольяттинский государственный университет
Danyaeared@gmail.com

На сегодняшний день становится популярным использовать технологии виртуализации рабочих столов. Большое количество компаний различных сфер деятельности стремится добиться утилизации вычислительных мощностей за счет технологии виртуализации рабочих столов. Но в образовательной сфере в большинстве случаев используется стандартная архитектура: компьютерные классы с персональными компьютерами. Стандартная архитектура обладает рядом существенных недостатков: