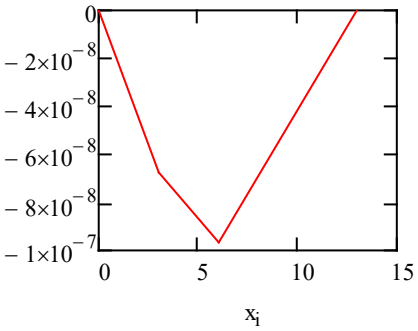
$$rez = Find\left(y_1, y_2, dy_0, dy_3\right)$$

$$rez = \begin{pmatrix} -6,746\cdot10^{-8} \\ -9,376\cdot10^{-8} \\ 2,491\cdot10^{-8} \\ 2,11\cdot10^{-8} \end{pmatrix}$$

$$y = \begin{pmatrix} y_0 \\ rez_0 \\ rez_1 \\ y_3 \end{pmatrix} \qquad y = \begin{pmatrix} 0 \\ -6,746\cdot10^{-8} \\ -9,376\cdot10^{-8} \\ 0 \end{pmatrix}$$



Таким образом, в конце расчетов по методу начальных параметров и методу конечных элементов мы свели к построению графиков, которые позволили нам визуально представить воздействие теплового поля на проводник и тем самым изначально учесть при проектировании слабые участки.

УДК 681.3

ОСОБЕННОСТИ ОРГАНИЗАЦИИ МОДЕЛИРОВАНИЯ СТОХАСТИЧЕСКИХ СЕТЕЙ В МНОГОЗАДАЧНЫХ СРЕДАХ

Воронов В.П., Муравьев Г.Л.
УО «Брестский государственный технический университет», г. Брест

Одной из задач, решаемых при программной организации имитационного моделирования, является реализация квазипараллельного выполнения процессов, когда исполнение процессов параллельно на уровне модели, но последовательно на уровне аппаратного обеспечения.

Для организации параллельных вычислений используют параллельные языки программирования (например, Modula, Ada, Algol-68) либо языки, расширенные соответствующими высокоуровневыми библиотеками и интерфейсами с переносом конструкций параллелизма с языкового уровня на уровень операционной системы (ОС) [1, 2].

Современные многозадачные ОС позволяют представлять выполняемые задачи в виде частей (нитей), выполняемых параллельно и взаимодействующих в едином глобальном пространстве. Пользователю при этом доступны развитые средства управления взаимодействием задач и нитей. Это специализированные библиотеки (например, MPI, PVM; классы объектно-ориентированной библиотеки MFC для управления потоковой многозадачностью системы Visual Studio C++), ориентированные на использование процессоров в режиме MIMD-машины, позволяющие управлять потоками, синхронизацией потоков, семафорами.

Представляется целесообразным реализацию параллельного выполнения процессов модели возложить на многозадачную ОС [3]. Это позволит повысить прозрачность исполнимого кода, упростить реализацию модельных описаний и, с учетом управления ОС переключением процессов на уровне блоков процессора, а также с учетом тенденции производителей процессоров для персональных ЭВМ к развитию многоядерной архитектуры, снизить трудоемкость реализации моделей.

В работе указанная задача рассматривается применительно к организации имитационного моделирования стохастических сетевых моделей (ССМ), рассматриваемых как совокупность управляющих и обслуживающих узлов и потоков обслуживаемых заявок [3]. Для распараллеливания вычислений используется параллелизм задач.

Задача сводилась:

- к анализу возможностей операционных систем в части распараллеливания;
- выбору вариантов декомпозиции модели на параллельно-функционирующие компоненты, представимые параллельными компонентами операционной системы;
- разработке алгоритмов и способов отображения имитационных моделей ОС-компонентами (включая связывание компонентов в единую структуру, обеспечение их информационного обмена, расчет характеристик модели и т.п.).

В качестве многозадачных ОС рассматривалось семейство систем UNIX (стандарт POSIX) и Unix-подобные системы, где работа с нитями определена стандартами (POSIX 1003.1c-1995, POSIX 1003.1-1988) и поддерживается соответствующими библиотеками типа Pthreads [4].

В окончательном варианте в качестве программной платформы для реализации моделирования была выбрана Unix-подобная ОС FreeBSD 7.2, отличающаяся полной реализацией системных вызовов, регулируемых стандартом POSIX.

Указанный стандарт регламентирует реализацию ряда объектов. Это мьютекс – простейший двоичный семафор, который может находиться в отмеченном (открытом) или неотмеченном (закрытом) состоянии, связывается и управляется нитью процесса (владельцем мьютекса). Это условная переменная - объект, который может использоваться для оповещения одной приостановленной нити другой нитью о наступлении некоего условия, после чего нить может продолжить работу.

В качестве перспективных подходов к организации моделирования предлагаются алгоритмы: на базе мьютексов и условных переменных; на базе конвейеров ОС. В работе выделены и рассмотрены особенности двух способов декомпозиции модели на нити: по потокам обслуживаемых заявок и по обслуживающим узлам.

В первом случае предполагается транзактное представление параллельностей в модели (в отдельную нить объединялись узлы, участвующие в обслуживании заявок одного класса), что характерно для функционального описания модели. Данный способ оправдывает себя при отсутствии общих узлов, через которые проходят заявки разных классов. При их наличии сложность алгоритма неоправданно растет из-за управления синхронизацией потоков заявок.

Во втором случае предлагается процессное представление параллельностей в модели (в отдельную нить выделяется функционирование каждого узла), что характерно для структурного описания модели. Здесь средства синхронизации нитей выделяются в отдельную сущность – канал, рассматриваемый как общая для нескольких нитей область памяти. Корректность работы с данными в этой области гарантируется использованием встроенных средств синхронизации.

Ниже приведен упрощенный алгоритм работы канала, связывающего нить-источник А и нить-приемник Б, обеспечивающий бесконечную передачу данных только в одну сторону (полудуплекс), что достаточно для моделирования ССМ [5]. Пусть канал инициализирован и нити запущены.

Тогда алгоритм передачи данных нитью А включает шаги: 1. Сгенерировать данные. 2. Открыть мьютекс mutex. 3. Загрузить данные в область buf. 4. Установить флаг ready. 5. Послать сигнал через cond. 6. Закрыть мьютекс. 7. Вернуться на пункт 1.

Алгоритм приема данных нитью Б включает шаги: 1. Открыть мьютекс. 2. Ждать, пока не придет сигнал через cond и не будет установлен флаг ready. 3. Обработать данные из buf. 4. Сбросить флаг ready. 5. Закрыть мьютекс. 6. Вернуться на пункт 1.

Здесь канал информационно отображается атрибутами: cond - условная переменная, информирующая нить Б об изменении состояния канала нитью А; mutex – мьютекс для защиты переменной cond; ready – флаг (признак) освобождения канала для защиты от некорректной посылки сигналов; buf – область памяти, общая для А и Б.

Заявкам стохастической сети прикрепляется модельное время, устанавливаемое генератором сети, которое модифицируется при прохождении каждого узла. При этом узел для каждой заявки сохраняет ряд атрибутов, включая модельное время входа и выхода заявки из узла, необходимых для проведения моделирования и для вычисления характеристик модели.

При использовании для процессного представления параллельностей конвейеров ОС предлагается каждый блок ССМ отображать отдельной программой, передающей и принимающей заявки в виде текстовых строк через стандартные потоки ввода-вывода, а узлы связывать с помощью конвейера ОС (например, на языке shell) [5]. Это позволит передавать данные из процесса-источника в процесс-приемник в виде строк, а синхронизацию процессов возложить на ОС. Для отображения маршрутных узлов ССМ предлагается использовать именованные конвейеры, представляемые файлами.

Таким образом, проведено исследование и макетирование алгоритмов и подходов к имитации стохастических сетевых моделей на языках С и shell с использованием библиотек ОС Unix. Установлено, что наибольшая эффективность моделей достигается при использовании процессного подхода, реализуемого на базе нитей библиотеки Pthread. При этом инструменты shell рекомендуется использовать на этапе прототипирования, т.к. здесь модели на порядок медленнее из-за использования операций ввода-вывода.

Литература

1. Воеводин, В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб.: BHV-Петербург, 2002. – 609 с.

2. Одинцов, И.О. Профессиональное программирование. Системный подход / И.О. Одинцов. - СПб.: BHV-Петербург, 2002. – 512 с.

3. Воронов, В.П. Имитационное моделирование в многозадачных ОС // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях: материалы XII респ. науч. конф. студентов и аспирантов, Гомель, 16-18 марта 2009 г.: в 2 ч. / ГГУ им. Ф.Скорины; редкол.: О.М. Демиденко [и др.]. – Гомель, 2009. – Ч. 2. – С. 17–18.

4. Многопотоковое программирование. – Режим доступа: http:// rk6.bmstu.ru/electronic_book/ progrws/multithread.html. – Дата доступа: 25.01.2008.

5. Стивенс, У.Р. UNIX: взаимодействие процессов / У.Р. Стивенс, – СПб.: Питер, 2003. – 188 с.