APPLICATION OF OBJECT-ORIENTED APPROACH FOR IMPLEMENTATION OF IMITATING MODELING OF QUEUING SYSTEMS

Ya. Zayachuk, Candidate of Technical Sciences, Associate Professor, Lecturer
E. Moyseenko, Candidate of Technical Sciences, Associate Professor, Lecturer
V. Kropyvnitskaya, Candidate of Technical Sciences, Associate Professor, Lecturer
Ivano-Frankivsk National Technical University of Oil and Gas, Ukraine

Authors have developed software of imitating modeling of processes performance in a computer system allowing to explore processes according to the stated scheduling algorithms.

Keywords: queuing systems, imitating modeling, scheduling algorithms.

Conference participants, National championship in scientific analytics

ПРИМЕНЕНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА ДЛЯ РЕАЛИЗАЦИИ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

Заячук Я.И., канд. техн. наук, доцент
Моисеенко Е.В., канд. техн. наук, доцент
Кропивницкая В.Б., канд. техн. наук, доцент
Ивано-Франковский национальный технический университет нефти и газа, Украина

Разработано программное обеспечение имитационного моделирования поведения процессов в компьютерной системе, которое дает возможность исследовать процессы в зависимости от заданных алгоритмов диспетчеризации.

Ключевые слова: системы массового обслуживания, имитационное моделирование, алгоритмы диспетчеризации.

Участники конференции, Национального первенства по научной аналитике

Компьютерные системы постоянно развиваются и становятся более мощными. В то же время расширяется набор задач, которые они решают, растет их количество и потребность в системных ресурсах. Имитационное моделирование является мощным инструментом, который позволяет определить оптимальные стратегии диспетчера для определенного набора процессов. Такое моделирование позволяет получить более детальное представление о поведении процессов в системе, чем прогнозированные расчеты и разработка аналитической модели.

Несмотря на значительный интерес к проблеме обслуживания процессов в компьютерных системах, особенно при разработке и реализации операционных систем, вопросу имитационного моделирования поведения процессов в системе большого внимания не уделялось [1-7]. Диспетчеризация является одной из ключевых концепций многозадачности, функционирования систем общего назначения и систем реального времени. Любая компьютерная система должна распределять свои ресурсы между многими процессами с потенциально конкурирующими требованиями. Существует лишь несколько программных продуктов, используемых для этой цели: MOSS, CPUSS, Cheddar и др. [1-7]. К недостаткам данных систем моделирования можно отнести: небольшое число алгоритмов диспетчеризации, работу которых можно сымитировать; невозможность настройки пользователем параметров этих алгоритмов; низкое количество статистических показателей, характеризующих работу системы; неразвитый и неудобный графический интерфейс. Кроме того, ни один из программных продуктов не позволяет провести имитационное моделирование поведения процессов в многопроцессорной системе. Поэтому актуальной и целесообразной является разработка системы имитационного моделирования, которая должна осуществлять моделирование инициализации обслуживания процессов с использованием заданных алгоритмов диспетчеризации, а также выводить статистические показатели и графические результаты. Интерфейс программы должен обеспечивать проведение исследований и анализ результатов без применения дополнительных средств программирования [7].

Согласно с вышеизложенными требованиями создано программное обеспечение для моделирования процессов диспетчеризации в компьютерных системах. Данный продукт реализован с помощью языка программирования C++ с использованием кросс-платформенного инструментария разработки Qt версии 4.6.0. При разработке программы соблюдались основные принципы объектно-ориентированного подхода [7].

Программа состоит из нескольких классов, объекты которых, в процессе ее выполнения, взаимодействуют между собой. Условно их можно разделить на классы графического интерфейса, используемые непосредственно для взаимодействия с пользователем, и вычислительные классы, созданные для функционирования самого ядра программы.

Программа позволяет сымитировать поведение системы массового обслуживания, а именно компьютерной системы с заданным количеством обслуживающих устройств (процессоров), обрабатывающей некоторое множество задач (процессов), согласно с указанными алгоритмами диспетчеризации. Процессы, поступающие в систему на выполнение, характеризуются определенными свойствами: скоростью поступления в систему и средним временем обслуживания. Программа имитирует обработку данных процессов с такими входными характеристиками в зависимости от заданных пользователем указаний и выводит статистические результаты их выполнения.

Программа моделирования процессов диспетчеризации содержит следующие классы:

SchedulerForm - класс графического интерфейса программы для работы с пользователем;

Scheduler - центральное вычислительное ядро программы с методами-алгоритмами диспетчеризации;

ProcessParameters - описывает параметры процессов в системе;

QueueParameters - описание очередей в системе;

CpuParameters- описывает процессоры системы;

Классы ProcessParameters, QueueParameters и CpuParameters объявлено дружественными классу Scheduler. Это сделано для того, чтобы методы последнего имели полный доступ к закрытым элементам этих классов. Таким образом, дружеские классы, которые не имеют собственных методов, фактически являются структурами класса Scheduler.

Класс Scheduler является основным вычислительным классом программы. Фактически он представляет собой класс краткосрочного планировщика, имитирующий выполнение процессов, распределяя процессорные ресурсы согласно заданному алгоритму диспетчеризации.

Объекты этого класса создаются методами класса графического интерфейса SchedulerForm, которым собственно и управляет пользователь, задающий входные характеристики системы.

При создании объект класса Scheduler получает информацию о количестве процессов в системе, которые необходимо обработать (mProcessAmount), заданное количество процессоров (mCpuAmount) и очередей (mQueueAmount). Данный класс содержит указатели на элементы дружественных ему классов ProcessParameters * process, CpuParameters * cpu и QueueParameters ** queue. К закрытым членам класса также относят заданные пользователем скорость поступления процессов в систему (mAvgArrivalSpeed) и среднее время выполнения процесса (mAvgBurstTime).

Конструктор класса Scheduler, используя входные параметры: заданное пользователем количество процессов, процессоров и очередей, а также указатели на дружеские классы ProcessParameters, QueueParameters и CpuParameters динамически создает объекты этих классов.

Метод RandomizeData (), используя введенные пользователем скорость поступления процессов в систему, тип распределения этих интервалов поступления, среднее время обслуживания процесса и его распределение, генерирует для каждого процесса случайные величины времени поступления в систему и необходимого времени обслуживания с заданными распределениями. Он работает по следующей схеме:

- вычисляется средний интервал времени между поступлениями процессов в систему, обратно пропорционален скорости поступления;
- время поступления в систему каждого процесса определяется как сумма времени поступления предыдущего процесса и среднего интервала времени между поступлением (для детерминированного типа распределения). При выборе пуассоновского распределения величина интервала генерируется с помощью генератора случайных чисел с экспоненциальным распределением, входным параметром для которого является величина среднего интервала;
- определяется необходимое время обслуживания для каждого процесса по заданным средним значениям для детерминированного распределения, и сгенерированным как в предыдущем случае значением для экспоненциального распределения.

Метод InitializeData() инициализирует процессы в системе, присваивая им идентификаторы и указатели на следующие процессы.

Методы диспетчеризации RunFcfs(), RunRr(), RunSjf(), RunPp(), RunMfq() работают с инициализированными объектами процессов, процессоров и очередей согласно своих алгоритмов.

К атрибутам класса относятся статистические показатели, характеризующие работу алгоритмов диспетчеризации: среднее время оборота процесса (avgTurnaroundTime), среднее время ожидания (avgWaitingTime) и нормализованное время оборота (normTurnaroundTime). Метод CalculateStatistics (), используя информацию о выполненных процессах, а методы GetData () и GetCriteria () используются для передачи полученных данных к объекту класса SchedulerForm и последующего их вывода. Метод CalculatePlotStats() служит для сбора статистической информации о процессах в форме удобной для ее графического представления. Он работает с переменной mPercentiles, методом для быстрой сортировки массивов QuickSort() и указателями **mpBurstTimes, *mpWaitingTimes, *mpNormTurnaroundTimes на массивы, которые динамически создаются конструктором класса Scheduler.

Коротко опишем работу данного метода: процессы с помощью функции QuickSort () группируются в перцентили (mPercentiles) в соответствии со временем их обслуживания. Часть процессов с минимальным временем обслуживания попадает в первый перцентиль, такая же часть процессов за исключением предыдущих попадает во второй перцентиль, и т.д. Вычисляется для каждого перцентиля среднее значение времени ожидания процесса и нормализованного времени оборота, хранящиеся в массивах mpWaitingTimes и mpNormTurnaroundTimes, соответственно. Метод GetPlotStats() передает полученные результаты к объекту класса SchedulerForm.

Класс ProcessParameters характеризует процессы, находящихся в системе.

Каждый объект этого класса - это процесс, который описывается определенным набором параметров, которые условно можно разделить на две группы. Входные (системные) - непосредственно характеризующие процесс при его инициализации, и выходные (статистические), которые будут использоваться при дальнейшем исследовании алгоритмов диспетчеризации. К первой группе относятся:

- идентификатор процесса (id);
- приоритет процесса (priority), если он указан пользователем;
- время вхождения в систему, т.е. инициализации (arrivalTime);
- время, необходимое процессу для выполнения (burstTime);
- состояние процесса (state);
- указатель на следующий инициализированный процесс (* next).

Статистические параметры, исчисляемых в ходе выполнения процесса, следующие:

- время начала обработки процесса (cpuArrivalTime)
- оставшееся время работы процесса до завершения (remainBurstTime)

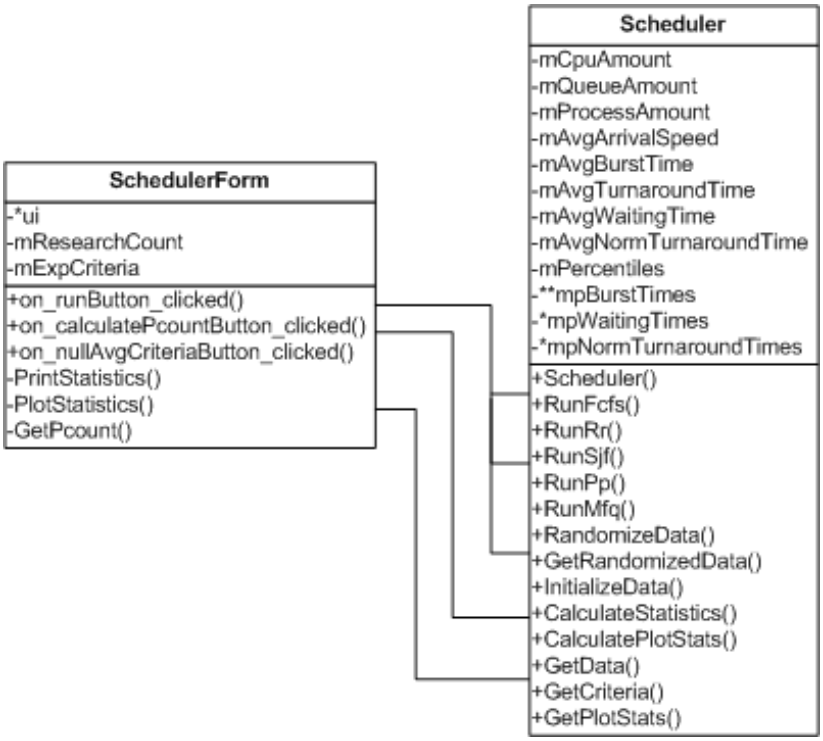

Рис. 1. Взаимодействие методов и атрибутов классов *Scheduler*, *ProcessParameters*, *QueueParameters*, *CpuParameters*

- время завершения процесса (finishTime)
- время полного оборота процесса (turnaroundTime)
- время ожидания (waitingTime).

Каждая из очередей, задаваемая пользователем, описывается с помощью класса QueueParameters, который содержит указатель на элемент очереди (* element). Класс параметров процессоров CpuParameters содержит указатель на процесс, который обрабатывается (* onCPU), а также квант времени (timeSlice), если он указан пользователем, отведенный на выполнение каждого из процессов. Обобщенную схему вышеописанного взаимодействия методов и атрибутов классов Scheduler, ProcessParameters, QueueParameters и CpuParameters возможно изобразить следующим образом (рис.1).

Класс SchedulerForm используется для интерактивной работы с пользователем. Объектом этого класса являются форма графического интерфейса программы. С ее помощью пользователь вводит характеристики системы для дальнейшего моделирования, дает указания относительно используемых алгоритмов, получая в результате необходимые статистические и графические данные.

Методы класса *SchedulerForm* используются в основном для вывода полученных результатов моделирования объектами класса *Scheduler*. Основные методы и атрибуты обеих взаимодействующих классов, отображены на рис. 2. Так метод *PrintStatistics(),* получая данные от объектов класса *Scheduler* через его методы *GetData()* і *GetCriteria()*, выводит характеристики процессов и статистические критерии алгоритмов диспетчеризации.

Метод PlotStatistics() используется для вывода графиков, которые отображают влияния стратегий планирования выполнения процессов. Он получает данные с помощью метода GetPlotStats() объекта класса Scheduler.

Для получения информации о количестве процессов в очереди и количество выполненных процессов в некоторый, заданный пользователем, момент времени существует метод GetPcount ().

Несмотря на случайное генерирование входных данных иногда при проведении эксперимента необходимо повторить моделирования несколько раз, чтобы добиться некоторого устойчивого результата, а именно средних значений статистических показателей для каждого из алгоритмов диспетчеризации. Так, переменная mResearchCount используется для подсчета количества проведенных моделирований, а полученные средние значения критериев хранятся в массиве mExpCriteria. Вывод данного массива значений выполняет вышеупомянутая функция PrintStatistics().

Для интерактивного взаимодействия с пользователем класс SchedulerForm имеет специальные методы, т.н. слоты. В среде Qt слоты - это функции, которые реагируют на сигналы пользователя и выполняют нужные действия.

Так, например, слоты on_configToolButton_clicked(), on_statsToolButton_clicked() и on_graphToolButton_clicked() используются для осуществления переходов между секциями программы. А слот on_nullAvgCriteriaButton_clicked() - для обнуления средних значений статистических показателей алгоритмов диспетчеризации. Слот on_runButton_clicked() является основным связующим методом объектов класса Scheduler и объектов формы графического интерфейса. Опишем его работу более подробно, поскольку он является одной из ключевых частей программы.

После нажатия кнопки запуска программы моделирования, данный слот получает введенные пользователем входные данные из формы.

Создается прототипный объект класса Scheduler, конструктор которого в качестве входных данных получает заданное пользователем количество процессов, процессоров и очередей системы.

Данный объект обрабатывается вышеописанным методом своего класса RandomizeData(), использующий введенные пользователем скорость поступления процессов в систему, среднее время обслуживания процесса и типы распределения этих величин.

В зависимости от заданных пользователем алгоритмов диспетчеризации, создаются дополнительные объекты класса Scheduler, по одному для каждого из алгоритмов.

С помощью метода GetRandomizedData() эти объекты получают от прототипа идентичные наборы сгенерированных методом RandomizeData() данных, далее к ним применяется метод InitializeData().

Объекты класса Scheduler обрабатываются согласно своих алгоритмов диспетчеризации RunFcfs() - алгоритм «FIFO», RunRr() - алгоритм кругового планирования, RunSjf() - алгоритм выбора кратчайшего процесса, RunPp() – алгоритм планирования с предпочтениями или RunMfq() - алгоритм многоуровневой очереди с обратной связью.

Методы класса SchedulerForm PrintStatistics() и PlotStatistics(), используя ссылку на обработанные объекты, получают и выводят результаты моделирования.

Алгоритмы диспетчеризации RunFcfs(), RunRr(), RunSjf(), RunPp(), RunMfq () являются методами класса Scheduler и начинают работу с его объектами после инициализации входных данных. Алгоритмы работают с объектами классов ProcessParameters, QueueParameters и CpuParameters, т.е. процессами, очередями и процессорами.

Программное обеспечение имитационного моделирования поведения процессов в компьютерной системе дает возможность исследовать процессы в зависимости от заданных алгоритмов диспетчеризации. Благодаря этому можно определить оптимальные характеристики и стратегии диспетчера системы для работы с заданным набором процессов. Использование парадигмы объектно-ориентированного программирования позволяет представить каждый элемент системы отдельным объектом, а функционирование этой системы - их взаимодействием. Благодаря открытости и бесплатности среды Qt, в которой создавалась программа, существует возможность разработки дополнительных или усовершенствование существующих модулей.

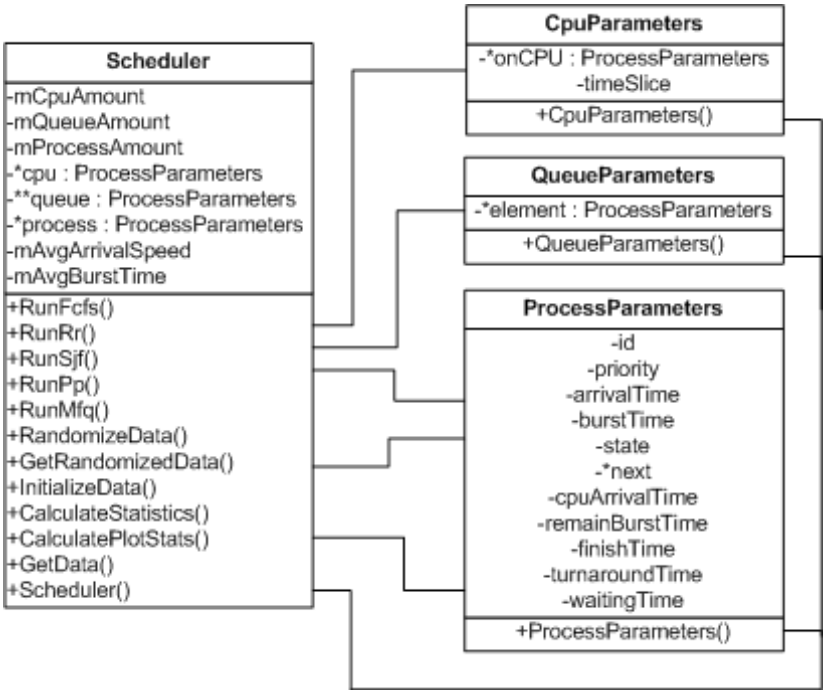

Рис. 2. Взаимодействие методов и атрибутов классов *Scheduler* та *SchedulerForm*

References:

1. Емельянов, В.В. Имитационное моделирование систем / В. В. Емельянов, С.И. Ясиновский. - М.: МГТУ им. Н.Э. Баумана, 2009. - 584 с. - ISBN 978-5-7038-3238-7.

2. Теории Случайных Функций, Марковских Процессов, Массового Обслуживания, Надежности и Восстановления в Приложении к Технической Эксплуатации Автомобилей / Владимиров Александр Федорович. - М.: Феникс, 2006. - 1 с. - ISBN 978-5-222-19438-6.

3. Афонин В.В. Моделирование систем / В.В. Афонин, С.А. Федосин. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2010. - 232 с. - ISBN 978-5-9963-0352-6.

4. Шелухин О.И. Моделирование информационных систем / О.И. Шелухин. - М.: Горячая Линия - Телеком, 2011. - 536 с. - ISBN 978-5-9912-0193-3.

5. Морозов В.К. Моделирование информационных и динамических систем / В.К. Морозов, Г.Н. Рогачев. - М.: Академия, 2011. - 384 с. - ISBN 978-5-7695-4221-3.

6. Колесов Ю.Б. Моделирование систем. Динамические и гибридные системы / Ю.Б. Колесов, Ю.Б. Сениченков. - М. : БХВ-Петербург, 2006. - 224 с. - ISBN 5-94157-578-5.

7. Сирота, А. Компьютерное моделирование и оценка эффективности сложных систем / А. Сирота. - М.: Техносфера, 2006. - 280 с. - ISBN 5-94836-080-6.