УДК 519.683

МЕТОДИКА ИСПОЛЬЗОВАНИЯ МНОГОПОТОЧНОГО ПРОГРАММИРОВАНИЯ ДЛЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ МНОГОПРОЦЕССНЫХ ГАЛЬВАНИЧЕСКИХ ЛИНИЙ

© 2013г. ***А.А. Евдокимов***

Тамбовский государственный технический университет

a_a_ewdokimow@mail.ru

Поступила в редакцию 30.11.2012

Рассмотрены основные особенности многопоточного программирования для осуществления параллельных вычислений. Представлена методика использования многопоточного программирования для автоматизированных систем многопроцессных гальванических линий с программным управлением. Приведен пример использования методики для разработки программного комплекса автоматизированной системы многопроцессной однорядной гальванической линии цинкования на подвесках.

Ключевые слова: автоматизированные системы, многопроцессные гальванические линии, параллельные вычисления, многопоточное программирование, потоки.

Введение

Объектом исследования являются автоматизированные системы многопроцессных гальванических линий с программным управлением.

Автоматизация промышленных производств происходит значительно медленнее, чем в иной другой сфере. Это связано с более высоким уровнем задач и высокой стоимостью времени нахождения оптимальных решений. Кроме того, производства испытывают необходимость не только в сборе, анализе и обработке поступающей информации от объектов, систем и процессов, но и в формировании точного, своевременного и эффективного решения для изменения состояния тех же объектов, систем и процессов, то есть управляющего воздействия. Автоматизированные системы управления широко используются для поддержания технологического цикла при организации поточных линий, диспетчеризации процессов производства на станочном оборудовании, диспетчеризации вспомогательными системами при производстве (освещением, вентиляцией, водоснабжением, электроснабжением, водоочисткой и т.д.).

Большинство автоматизированных систем получает, анализирует и обрабатывает информацию от измерительных устройств (датчиков). Количество датчиков напрямую влияет на производительность системы. При сборе, анализе и обработке информации тратятся временной и вычислительный ресурсы. Для снижения времени обработки и увеличения производительности необходимо использовать параллельные вычисления. Такой подход позволит также автоматизированным системам управления быстрее принимать управляющее воздействие и эффективнее решать свои задачи.

Целью распараллеливания, как и большинства методов программирования, является оптимальное использование системных ресурсов. Параллельная обработка повышает сложность проектирования, тестирования и сопровождения программ, обеспечивая при этом повышение пропускной способности приложений на одно- и многопроцессорных компьютерах. Многопоточность представляет собой значительный шаг вперед по сравнению со временами, когда параллельная обработка осуществлялась за счет взаимодействия процессов. За счет распараллеливания сложность организации может быть несколько снижена, особенно в отношении взаимодействия процессов.

Потоки – это независимые друг от друга задачи, выполняемые в контексте процесса. Поток использует код и данные родительского процесса, но имеет свой собственный уникальный стек и состояние процессора, включающее указатель команд. Потоки действуют аналогично процессам и так же, как процессы, коллективно используют ресурсы центрального процессора (ЦП). Каждый поток – это отдельный канал управления, обеспечивающий независимое исполнение своих команд, что позволяет многопоточному процессу одновременно решать различные задачи [1].

Приложение, спроектированное и разрабо-

НИР выполнена в соответствии с проектом №14.B37.21.0176 Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг.

танное с использованием многопоточного программирования, имеет следующие преимущества:

- рост производительности за счет аппаратных средств, обеспечивающих многозадачность (параллелизм). Как правило, приложения, реализующие параллелизм через потоки, не должны учитывать число доступных процессоров. Производительность приложения равномерно увеличивается при наличии дополнительных процессоров. Численные алгоритмы и приложения с высокой степенью параллелизма могут выполняться намного быстрее;
- повышение пропускной способности приложения. Приложение, имеющее и оптимизированное для решения задачи некоторым количеством потоков, в отличие от последовательного приложения способно выполнить большее количество задач данного класса;
- повышение скорости реакции приложения. Любое приложение, содержащее много не зависящих друг от друга действий, может быть перепроектировано так, чтобы каждое действие выполнялось в отдельном потоке. Например, пользователь многопоточного интерфейса не должен ждать завершения одной задачи, чтобы начать выполнение другой;
- более эффективное использование системных ресурсов. Приложения, использующие два или более процессов, которые имеют доступ к общим данным через разделяемую память, содержат более одного потока управления. При этом каждый процесс имеет полное адресное пространство и состояние в операционной системе. Стоимость создания и поддержания большого количества служебной информации делает каждый процесс более затратным, чем поток. Кроме того, разделение работы между процессами может потребовать от разработчиков значительных усилий, чтобы обеспечить связь между потоками в различных процессах или синхронизировать их действия;
- оптимальная структура кода. Некоторые приложения более эффективно представляются в виде нескольких независимых или полуавтономных единиц, чем в виде единого монолитного приложения. Многопоточные приложения легче адаптировать к изменениям требований конечного пользователя.

1. Методика использования многопоточного программирования

Использование многопоточного программирования для автоматизированных информационных систем сводится к выполнению разработчиками программного обеспечения таких действий, как:

1) объектная декомпозиция автоматизированной информационной системы;

2) выбор метода распараллеливания;

3) функциональная и/или структурная декомпозиция;

4) выделение информационных взаимодействий;

5) выбор вычислительного оборудования;

6) разработка и оптимизация программного обеспечения.

Объектная декомпозиция автоматизированной информационной системы предназначена для выделения относительно самостоятельных элементов системы, участвующих в вычислительных схемах, а также их взаимосвязи. Объектная декомпозиция сводится к процессу представления предметной области задач автоматизированной системы в виде совокупности объектов, обменивающихся сообщениями. Выделенные объекты участвуют не только в обмене сообщениями, но и могут также воздействовать на состояние других объектов или испытывать определенное воздействие со стороны других объектов. Этот шаг необходим также для построения математической модели автоматизированной системы. Объектная декомпозиция особенно актуальна при использовании объектно-ориентированных языков программирования. Она обеспечивает возможность повторного использования программных модулей, снижение затрат на увеличение их функциональности и встраивания в другие программные комплексы.

Для эффективного распараллеливания приложения необходимо выбрать наиболее подходящий метод распараллеливания. Различия между методами распараллеливания не оказывают заметного влияния на производительность программного комплекса, однако неправильный выбор метода приводит к увеличению затрат времени на внесение изменений, отладку и настройку распараллеленного программного комплекса.

Для выбора оптимального метода распараллеливания следует описать программный комплекс с точки зрения двух моделей: модели параллельного выполнения задач и модели параллельного использования данных. В приложениях с параллельным выполнением задач независимые операции, заключенные в функции, переносятся в выполняемые асинхронно потоки. Для выражения параллелизма на уровне задач предназначены библиотеки поддержки много-

поточности, например, интерфейс программирования многопоточных приложений Win32. Под параллелизмом на уровне данных подразумевается многократное применение одних и тех же команд или операций к различным данным. Хорошими кандидатами на применение методов параллельного использования данных являются циклы, в которых выполняются интенсивные вычисления. Общим примером реализации параллелизма на уровне данных может служить типичный алгоритм обработки изображений, который применяет фильтр к одной или нескольким точкам изображения для вычисления нового значения для заданной точки. Поскольку операции, выполняемые над точками изображения, являются независимыми, вычисления новых значений для точек могут выполняться параллельно. Иногда параллелизм на уровне данных может выражаться компилятором автоматически. Можно также описать параллелизм с помощью синтаксиса директив, определенного стандартом OpenMP. Функции преобразования директив в параллельный код выполняет компилятор.

В действительности целевой платформой многих параллельных приложений является компьютер с одним процессором. Применение многопоточности обусловлено двумя основными требованиями: увеличением параллелизма и повышением производительности ПО.

Следующим шагом является осуществление функциональной или структурной декомпозиции автоматизированной информационной системы. На данном этапе возможно также совместное выполнение этих двух типов декомпозиций. Оно может быть обусловлено такими результатами выбора метода распараллеливания, когда для описания объектов рассматриваемой автоматизированной информационной системы подходит и модель параллельного выполнения задач, и модель параллельного использования данных. В случае, когда поведение и взаимодействия объектов описываются моделью параллельного выполнения задач, проводят функциональную декомпозицию автоматизированной информационной системы. Если при этом поведение объекта системы сводится к выполнению определенной функции, например, изменение состояния другого объекта системы, то оно формируется в виде подзадачи, которая может быть реализована независимо от других подзадач. В случае, когда поведение и взаимодействия объектов могут быть описаны моделью параллельного использования данных, проводят структурную декомпозицию автоматизированной информационной системы. При этом выделяются такие части информационных процессов в автоматизированной системе, выполнение которых занимает наибольшее время или их выполнение связано с однотипным итерационным изменением каких-либо структур данных [2].

Выделение информационных взаимодействий – выделение для сформированного набора подзадач информационных взаимодействия, которые должны осуществляться в ходе решения исходной поставленной задачи. Выполнение этого шага очень важно для обеспечения работоспособности проектируемого программного комплекса. Необходимо выделить не только полезные взаимодействия подзадач, то есть такие, которые способствуют достижению общего результата их совместных вычислительных действий, но и все возможные в процессе их взаимодействия взаимные блокировки и коллизии [3].

Выбор вычислительного оборудования сводится к определению необходимой (или доступной) для решения задачи вычислительной системы и выполнению распределения имеющего набора подзадач между процессорами (или ядрами процессора) системы;

Шаг, на котором производятся разработка и оптимизация программного обеспечения, должен учитывать результаты предыдущих действий. На этом шаге важно использовать всю полученную информацию о системе, которую предстоит автоматизировать. При создании программного комплекса можно использовать следующие методики распараллеливания: выполнение явного распараллеливания программы с помощью обращений к функциям интерфейса программирования многопоточных приложений современных языков программирования, описание параллелизма приложения с помощью псевдокомментариев или директив OpenMP.

2. Применение методики использования многопоточного программирования

В качестве информационной системы, для которой необходимо провести автоматизацию с использованием многопоточного программирования, была выбрана система автооператорной однорядной многопроцессной гальванической линии цинкования на подвесках. Количество автоматических операторов в данной системе – 2, количество температурных датчиков – 7, количество позиционных датчиков – 21 горизонтальных и по 2 вертикальных на каждом автоматическом операторе, количество датчиков занятости – 21, количество датчиков потока – 1.

Сбор информации осуществляется с позиционных датчиков автоматических операторов, датчиков занятости технологических операций, температурных датчиков технологических операций с регулируемой температурой, датчика потока воздуха в сушильной камере. Управляющие воздействия, формируемые разработанным программным комплексом, направлены на регулирование температурного режима в технологических операциях, эффективное и правильное последовательное выполнение автоматическими операторами перемещения носителей по технологическим операциям, сигнализация и предупреждение возможных аварийных ситуаций.

Проектирование и разработка программного комплекса заключалась в выполнении предложенной методикой использования многопоточного программирования последовательности действий.

В результате выполнения объектной декомпозиции определены следующие объекты рассматриваемой системы:

- автоматический оператор, способный осуществлять выполнение заданного маршрута движения, заложенного в циклограмме. Циклограмма – описание действий автооператора: горизонтальных перемещений по позиционным датчикам, соответствующим технологическим операциям, и вертикальных перемещений траверсы по позиционным датчикам, обеспечивающим беспрепятственный подъем и перенос технологических носителей. Автоматический оператор испытывает управляющие воздействия в зависимости от текущей команды циклограммы;
- технологический носитель (в рассматриваемой системе – подвеска). Последовательно переносится по заданным технологическим операциям с целью получить гальваническое покрытие требуемого качества;
- технологическая операция (гальваническая ванна). Характеризуется временем нахождения в ней технологического носителя, которое гарантируется и обеспечивается определенными перемещениями автоматических операторов, заложенных в циклограмме;
- позиционный датчик, в качестве которого используется бесконтактный индуктивный датчик. Принцип действия основан на изменении параметров магнитного поля, создаваемого катушкой индуктивности внутри датчика. Информацией от позиционного датчика является сигнал о наличии/отсутствии в данной позиции автооператора (или траверсы автооператора);
- температурный датчик, в качестве которого используется термометр сопротивления. Принцип действия основан на изменении электрического сопротивления в зависимости от температуры среды, в которую помещен датчик. Информацией от температурного датчика является значение температуры в той технологической операции, в которую установлен указанный датчик;
- датчик занятости, в качестве которого используется бесконтактный индуктивный датчик. Принцип действия аналогичен принципу действия позиционного датчика. Информацией от датчика занятости является сигнал о наличии/отсутствии технологического носителя в конкретной технологической операции;
- датчик потока воздуха, в качестве которого используется реле контроля потока. Принцип действия основан на обнаружении наличия потока воздуха с помощью лепестковой пластины, устанавливаемой в контролируемой среде. Информацией от датчика потока воздуха является сигнал о наличии/отсутствии потока воздуха в технологической операции (в рассматриваемой системе – сушильной камере);
- нагревательный элемент, в качестве которого используется электронагреватель. Является объектом, на который оказывается управляющее воздействие в зависимости от полученной от температурных датчиков информации;
- концевой выключатель. Формирует сигнал при возникновении следующих событий: столкновении автоматических операторов, выезде автоматического оператора за пределы обслуживаемой зоны;
- аварийный выключатель (аварийная кнопка). Формирует сигнал при нажатии о возникновении аварийной ситуации. Контроль нажатия осуществляется обслуживающим персоналом рассматриваемой автоматизированной системы.

Проектируемый программный комплекс описывается моделью параллельного выполнения задач, и для выделения подзадач отдельных элементов рассматриваемой автоматизированной системы необходимо выполнить функциональную декомпозицию.

Результатом выполнения функциональной декомпозиции стало выделение следующих подзадач, требующих распараллеливания:

- подзадача сбора, анализа информации с температурного датчика и выработка соответствующего управляющего воздействия на устройство регулирования температуры;
- подзадача сбора, анализа информации с позиционных датчиков автоматического оператора и выработка соответствующего управляю-

щего воздействия по перемещению в соответствии с заданной технологической схемой;

- подзадача сбора и анализа информации с датчика занятости технологической операции для корректировки управляющих воздействий по перемещению автоматических операторов;
- подзадача сбора и анализа информации с датчика потока для оповещения обслуживающего персонала и выработки управляющего воздействия на устройство регулирования температуры в сушильной камере;
- подзадача анализа состояния системы для определения работоспособности отдельных элементов, регистрирования момента отказа оборудования и устройств, оповещения обслуживающего персонала и выработки управляющего воздействия на прекращение работы системы, если дальнейшее ее функционирование не будет соответствовать требованиям производительности.

Также необходимо отметить, что каждая из выделенных подзадач должна обеспечить понятный, информативный, интерактивный графический интерфейс для обслуживающего систему персонала. Кроме того, каждая из выделенных подзадач должна предоставить программному модулю, отвечающему за ведение журнала работы системы, исчерпывающую информацию о каждом выполненном действии.

Результатом выполнения следующего шага предложенной методики стало выделение следующих информационных взаимодействий:

- взаимодействие подзадач сбора, анализа информации с позиционных датчиков автоматических операторов с целью синхронизации работы самих автооператоров, а также совместного использования каналов связи с преобразователями;
- взаимодействие подзадач сбора и анализа информации с датчиков занятости технологических операций с подзадачами сбора, анализа информации с позиционных датчиков автоматических операторов с целью корректировки управляющих воздействий по перемещению автоматических операторов;
- взаимодействие всех подзадач с графическими программными компонентами с целью обеспечения понятного, информативного и интерактивного графического интерфейса для конечного пользователя системы;
- взаимодействие всех подзадач с подзадачей анализа состояния системы с целью предоставления информации о тех элементах системы, за которые отвечает конкретная подзадача.

В качестве вычислительной системы, на базе которой будут выполняться перечисленные подзадачи, может быть использован любой промышленный компьютер.

На шаге разработки программного комплекса под выполнение каждой подзадачи выделен отдельный поток в соответствии с рис. 1. Для правильной организации вычислительного процесса всей системы наиболее внимательно проектировались и разрабатывались взаимодействия между подзадачами. Для их реализации были использованы программные средства синхронизации межпоточных коммуникаций [4].

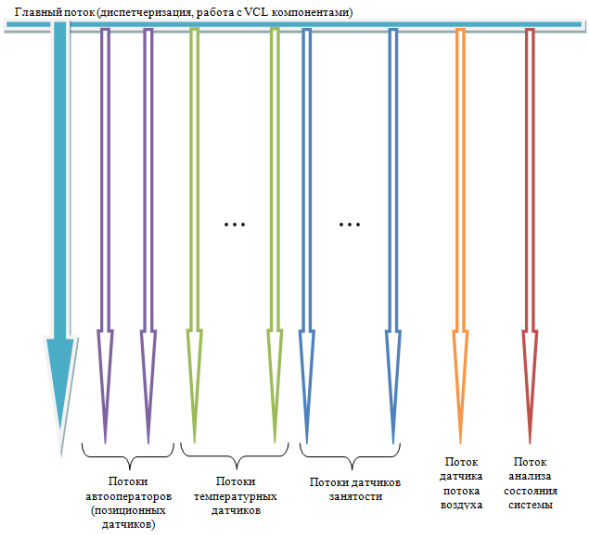

Рис. 1

В соответствии с результатами объектной и функциональной декомпозиции, а также выделенными информационными взаимодействиями, были спроектированы и реализованы следующие модули экспериментального программного комплекса:

- коммуникационный модуль, отвечающий за реализацию необходимых функций открытого промышленного протокола связи Modbus RTU;
- модуль ввода данных из конфигурационных файлов и файлов циклограмм, содержащих необходимую для работоспособности программного комплекса информацию о технологических процессах рассматриваемой автоматизированной системы;
- модуль инициализации исходных данных;
- модуль организации первичных настроек температурных режимов в соответствующих технологических операциях;
- модули диспетчеризации температурных режимов в технологических операциях;
- модули диспетчеризации работы автоматических операторов по осуществлению заданных технологических процессов;

- модули контроля занятости технологических операций;
- модуль контроля потока воздуха;
- модуль аварийной сигнализации, ведущий проверку состояния системы на возможность дальнейшего функционирования;
- модули графического интерфейса пользователя, реализующие наглядное и информативное отображение работы автоматизированной системы, возможность интерактивной корректировки температурных режимов, работы автоматических операторов;
- модуль ведения журнала работы системы;
- модуль вывода текущей информации о состоянии элементов системы во временные файлы, необходимой для корректного продолжения работы после возникновения аварийных ситуаций.

Для оценки производительности работы программного комплекса в зависимости от используемых вычислительных ресурсов использовался закон Амдала:

$$S_p = \frac{1}{\alpha + \frac{1-\alpha}{p}},$$

где α – время, затраченное на выполнение последовательных операций, а p – количество процессоров вычислительной системы. Время, использованное на служебные операции по организации потоков, не учитывается, поскольку создание пула потоков в рассматриваемой системе происходит единожды и в момент начала работы программного комплекса.

Согласно этому закону, ускорение выполнения программного комплекса за счёт распараллеливания инструкций его модулей на множестве вычислителей ограничено временем, необходимым для выполнения последовательных инструкций его модулей.

Таблица 1 показывает, во сколько раз быстрее выполнится программа с долей последовательных вычислений α при использовании p процессоров.

Таблица 1

$\alpha \setminus p$	10	100	1000
0	10	100	1000
10%	5.263	9.174	9.910
25%	3.077	3.883	3.988
40%	2.174	2.463	2.496

Из таблицы видно, что только модуль, вовсе не содержащий последовательных вычислений (α=0), позволяет получить линейный прирост производительности с ростом количества вычислителей в системе. Если доля последовательных вычислений в модуле равна 25%, то увеличение числа процессоров до 10 дает ускорение в 3.077 раза, а увеличение числа процессоров до 1000 даст ускорение в 3.988 раза.

Закон Амдала показывает, что прирост эффективности вычислений зависит от алгоритма задачи и ограничен сверху для любой задачи с $\alpha \neq 0$ в соответствии с рис. 2. Не для всякой задачи имеет смысл наращивание числа процессоров в вычислительной системе. Более того, если учесть время, необходимое для передачи данных между узлами вычислительной системы, то зависимость времени вычислений от числа узлов будет иметь максимум. Это накладывает ограничение на масштабируемость вычислительной системы, то есть означает, что с определенного момента добавление новых узлов в систему будет увеличивать время расчёта задачи.

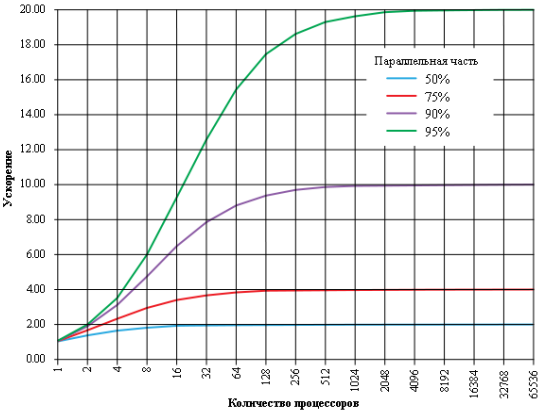

Рис. 2

Проведение экспериментов работы программного комплекса показало, что доля параллельных вычислений составляет около 95%, оставшаяся часть приходится на работу с VCL компонентами и организацию коммуникаций с преобразующими устройствами. Использовалось вычислительное оборудование с 4 процессорами, на котором получено ускорение 3.7, что оказалось достаточно для эффективной работы рассматриваемой автоматизированной системы.

Заключение

На основании проведенных исследований можно сделать следующие краткие выводы:

- современные информационные технологии позволяют автоматизировать информационные системы различной сложности;
- автоматизированные системы реализуют информационные процессы сбора, подготовки,

анализа, обработки, хранения, передачи информации, а также формируют управляющие воздействия на объекты, включенные в эти системы;

- использование параллельных вычислений не только естественно необходимо для реализации информационных процессов в автоматизированных системах, но и способствует увеличению их эффективности, производительности, достижению показателей более высокого качества;
- основные языки программирования представляют целый ряд средств и методов распараллеливания задач автоматизированных систем, синхронизации информационных взаимодействий при выполнении задач автоматизированных систем. Кроме того существуют специальные стандарты и платформы для организации параллельных вычислений;
- производители промышленного вычислительного оборудования предоставляют разработчикам широкие возможности для построения высокоэффективных программных комплексов автоматизированных систем.

Предложенная методика не претендует на окончательное и универсальное решение задачи использования параллельных вычислений для автоматизированных систем многопроцессных гальванических линий, и может дополняться этапами, необходимыми для решения прикладных задач каждой конкретной автоматизированной системы.

НИР выполнена в соответствии с проектом №14.B37.21.0176 Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг., реализуемого в рамках программы суперкомпьютерных работ Нижегородского университета». В ходе научно-исследовательской работы использовался подход, предложенный в [5].

Список литературы

1. Фоснер Р. Программирование с масштабируемой многопоточностью на основе пулов потоков. MSDN Magazine, 2010. URL: http://msdn.microsoft.com/ru-ru/magazine/ gg232758.aspx (дата обращения: 10.08.2012).
2. Методика разработки многопоточных приложений: принципы и практическая реализация. Intel, RSDN Magazine, 2004. URL: http://www.rsdn.ru/article/baseserv/ RUThreadingMethodology.xml (дата обращения: 05.08.2012).
3. Гергель В.П. Теория и практика параллельных вычислений. Интуит, 2007. URL: http://www.intuit.ru/department/calculate/paralltp/ (дата обращения: 29.07.2012).
4. Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования: пер. с англ. М: Издательский дом «Вильямс», 2003. 512 с.
5. Гергель В.П., Стронгин Р.Г. Опыт Нижегородского университета по подготовке специалистов в области суперкомпьютерных технологий // Вестник Нижегородского университета им. Н.И. Лобачевского. 2010. № 3-1. С. 191–199.

METHODOLOGY OF USE MULTITHREADED PROGRAMMING FOR AUTOMATED SYSTEMS MULTIPROCESS GALVANIC LINES

A.A. Evdokimov

The main features of multithreaded programming for parallel computing. Presents a methodology of multi-threaded programming for automated systems multiprocess galvanic lines with program management. An example of using methods for developing software complex of automated systems multiprocess-row galvanic zinc-plating on hangers.

Keywords: automated systems, multi-process galvanic lines, parallel computing, multi-threaded programming, threads.