УДК 735.29.(32)

УПРАВЛЕНИЕ РЕСУРСАМИ ПРИ ВЫПОЛНЕНИИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

Савченко Г.В.
Научный руководитель – д.т.н. Легалов А.И.
Сибирский федеральный университет

Разработка прикладных параллельных программ, как правило, характеризуется использованием особенностей вычислительных ресурсов конкретной архитектуры. Параллельные системы программирования выделяют множество процессов для выполнения прикладной задачи, а программист выстраивает их взаимодействия и взаимную передачу данных для решения последней, учитывая специфику платформы. Это обуславливается необходимостью более эффективного выполнения программы на некоторой архитектуре.

Во многих параллельных системах программирования используется архитектурно-независимый принцип, когда программист лишь определяет прикладной алгоритм, а внешние утилиты – компиляторы, компоновщики, виртуальные машины и операционные системы – обеспечивают связь программы с вычислительными ресурсами. Использование таких систем программирования обладает многими положительными сторонами, среди которых – возможность адаптации прикладной программы к другой архитектуре после завершения процесса отладки. К ним можно отнести функциональные языки программирования, например, Пифагор.

Для организации эффективной связи таких программ с вычислительными ресурсами представляется целесообразным выделить в отдельную задачу управление последними. В настоящее время отсутствуют методы описания ресурсов, не зависящие от используемого языка программирования. Поэтому актуальной проблемой является выделение процессов управления ресурсами и создание языка, позволяющего управлять вычислительными ресурсами и процессами параллельной программы. В рамках данной проблемы авторами решаются следующие задачи:

- описание специфики использования вычислительного ресурса и процесса обработки информации;
- определение состояний, в которых могут находиться ресурс и процесс;
- установление и управление связями ресурс-процесс при выполнении программы (управление ресурсами);
- использование языка управления ресурсами для описания алгоритма управления ресурсами – раздельно от прикладного алгоритма программы.

В качестве примеров управления ресурсами целесообразно привести алгоритм работы операционных систем, обеспечивающих управление ресурсами в мультипрограммной среде. Однако данные алгоритмы не обеспечивают гибкое управление ресурсами для одного параллельного процесса операционной системы. Еще один пример – полностью ручное управление ресурсами программистом при разработке приложений, например, на OpenMP, когда достигается повышение эффективности конкретной задачи. В таком случае код прикладного алгоритма смешивается с кодом управления ресурсами и организации вычислений, что приводит к архитектурной зависимости.

Для решения поставленных задач предлагается модель вычислительного ресурса и процесса. Под *вычислительным ресурсом* (далее просто ресурсом) подразумевается средство, предназначенное для выполнения вычислительных операций. Можно классифицировать ресурсы по типу выполняемых действий, но это ведет к усложнению мо-

дели. В данной работе под ресурсом можно понимать, например, процессорный модуль, способный выполнять команды из набора машинных инструкций компьютера.

Под *процессом* в работе подразумевается логически объединенная последовательность элементарных вычислений. Сама сущность параллельной программы располагает к формированию таких независимых порций вычислений.

Ресурсы и процессы в ходе выполнения программы могут находиться в различных состояниях. Множество состояний ресурса - R. Такие состояния называются операционными, потому что они связаны с выполнением программы, которая представляет собой набор действий (операций или операторов), а также зависят от времени. $R = \{R_f, R_c, R_l, R_w\}$.

R_f – (free) – ресурс свободен. Начальное состояние ресурса. Из данного состояния возможен переход в состояние R_c. На этом переходе ресурс блокируется в некотором глобальном пуле ресурсов, чтобы им не могли воспользоваться другие потребители.

R_c – (capture) – ресурс захвачен (заблокирован, занят). С этого момента он уже не может быть заблокирован другим потребителем. Из этого состояния, как правило, выполняется загрузка данных в ресурс и последующий переход в состояние R_l. Но возможна также разблокировка и переход в свободное состояние R_f.

R_l – (load) – данные в ресурс загружены. Данное состояние означает то, что аргументы и данные для выполнения процесса были успешно прочитаны ресурсом, перемещены в его внутреннюю память, и ресурс готов к выполнению. Из этого состояния возможен единственный переход в состояние функционирования R_w, когда ресурс занят вычислением.

R_w – (work) – ресурс функционирует, т.е. выполняет некоторое функциональное преобразование над загруженными в него данными. Из данного состояния возможен переход в R_c. На этом переходе происходит выгрузка или сохранение данных контекста. Выгрузка обязательна в связи с тем, что не завершивший функциональное преобразование ресурс можно остановить только в случае, если планируется назначить на него другую работу. В этом случае выгрузка данных обязательна, т.к. иначе данные могут быть «затерты» новой порцией работы.

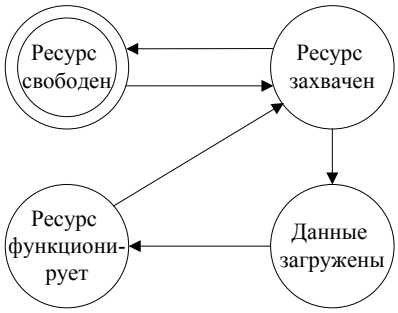

Рисунок 16. Операционные состояния ресурса

Множество возможных состояний процесса обозначается как $P = \{P_i, P_f, P_b, P_{pw}, P_{pr}, P_{pf}, P_a, P_l, P_w, P_r\}$.

P_i – (inactive) – процесс неактивен. Данное состояние означает, что существует некоторая порция вычислений, логически сгруппированная в процесс, аргументы для которой пока еще не заданы. Из данного состояния процесс может перейти в состояние P_{aw}.

P_{aw} – (activation waiting) – ожидание активации процесса. Данное состояние является композитным. На выходе из этого состояния процесс должен быть полностью

готов к выполнению, т.е. все предварительные условия, требуемые для его запуска, должны быть выполнены.

Под предварительными условиями запуска процесса понимаются:

1. аргументы для выполнения процесса прочитаны.
2. Ресурс для выполнения функционального преобразования процесса найден.
3. Ресурс эксклюзивно заблокирован и не может быть доступен другим процессам.

Подсостояния композитного состояния P_{aw}, которые показывают пути удовлетворения предварительных условий, показаны ниже (Рисунок 18). Предварительные условия (получение аргументов, поиск ресурса, доступность ресурса) могут удовлетворяться в различных последовательностях (например, сначала чтение аргументов, потом поиск ресурса и наоборот).

P_f – (find) – ресурс найден. Это означает, что для выполнения процессу подобран один из ресурсов, но это не означает, что выполнение будет продолжено именно в этом ресурсе. Далее выполняется попытка блокировки найденного ресурса. Если блокировка не удалась (например, данный ресурс уже заблокирован), то выполняется переход в состояние P_{fp}. В случае успешной блокировки ресурс блокируется, и выполняется переход в состояние P_b.

P_b – (blocked) – ресурс заблокирован. Это означает, что ресурс окончательно занят текущим процессом, и дальнейшее выполнение последнего произойдет именно в найденном ресурсе. Далее выполняется выход из композитного состояния P_{aw}.

Остальные состояния (P_{pw} – ожидание аргументов, P_{pr} – аргументы получены, P_{pf} – поиск ресурса) композитного состояния P_{aw} возникают при удовлетворении вышеперечисленных предварительных условий запуска процесса.

P_a – (active) – процесс активен. Из данного состояния возможна загрузка данных процесса в связанный ресурс и переход в состояние P_l.

P_l – (load) – данные процесса загружены в ресурс. Это состояние означает, что данные были успешно прочитаны ресурсом, и ресурс готов к выполнению функционального преобразования процесса. Из этого состояния возможен переход в состояние P_w, когда ресурс занят вычислением.

P_w – (work) – процесс выполняется в ресурсе. Ресурс производит вычисления, определяемые функциональными преобразованиями процесса и записывает их результаты в область памяти, выделенной для этого.

Возможны два варианта:

1. Когда съем данных процесса не требуется, происходит освобождение ресурса, который был эксклюзивно захвачен, и переход в неактивное состояние P_i.
2. В противном случае процесс переходит в состояние P_r.

P_r – (retrieve) – ожидание съема данных. В данном состоянии процесс ожидает, когда подсистема управления считает результат его работы, после чего процесс переходит в неактивное состояние P_i. Диаграмма перехода между состояниями процесса приведена ниже (Рисунок 17).

Приведенные модели состояний ресурсов и процессов могут быть преобразованы в автоматную модель. Для этого каждое состояние должно иметь дуги перехода на себя, потому что предполагается, что автомат меняет свое состояние с новым тактом дискретного времени, а процессы и ресурсы могут находиться в своих состояниях до срабатывания предикатов перехода. На рисунке дуги перехода на себя не показаны с целью упрощения восприятия.

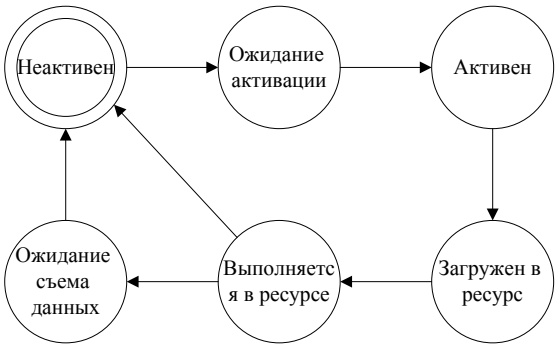

Рисунок 17. Операционные состояния процесса

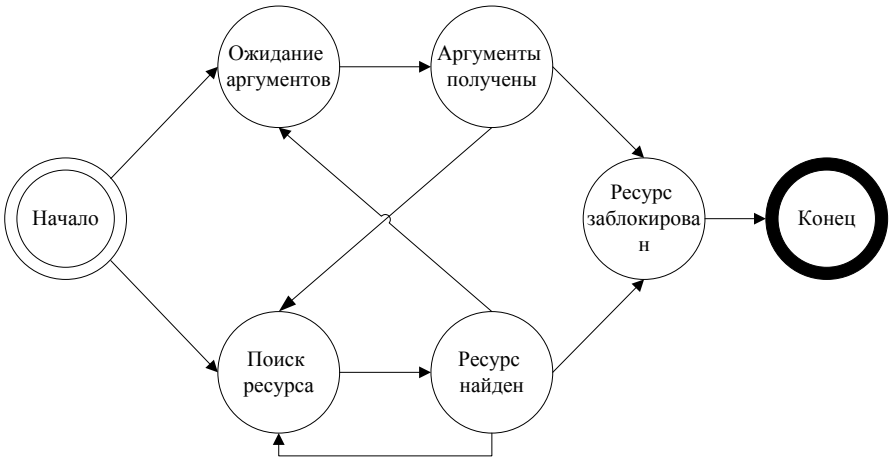

Рисунок 18. Подсостояния композитного состояния процесса "Ожидание активации", P_{aw}

Для решения задачи управления ресурсами предлагается ввести язык управления ресурсами (ЯУР), который должен содержать операторы для смены состояний ресурса и процесса, а также способ задания предикатов перехода между состояниями. Этот язык будет интерпретироваться системой управления ресурсами.

Задание стратегии на ЯУР представляет собой способ добавить «статичности» описания ресурсов в прикладную параллельную программу. Перенос программы на другую архитектуру выполняется автоматически. В этом случае на «новой» архитектуре будет использоваться автоматически генерируемая динамическая стратегия управления ресурсами. Если скорость или какая-либо другая характеристика программы неудовлетворительна, программист имеет возможность добиться требуемых показателей путем разработки стратегии для «новой» архитектуры.

Предлагаемая модель организации управления ресурсами представляет собой подход к эффективной интерпретации ресурсно-независимых программ путем раздельного описания прикладного алгоритма и ресурсов для его выполнения. С этой целью в работе приведена модель ресурсов и процессов обработки информации, рассмотрены возможные состояния и переходы между ними.