ГЛАВА 1

МЕТОД СТРУКТУРИРОВАНИЯ ПРОГРАММНЫХ РЕСУРСОВ

1.1. Конструктивные элементы параллельных систем

Конструктивными элементами для построения математических моделей функционирования программ, реализующих методы параллельной обработки решения задач, являются понятия *процесса* и *программного ресурса*.

Понятие *процесса* является одним из основополагающих в теории и практике параллельного программирования. В литературе разброс в трактовке данного понятия является достаточно широким, но в целом большинство определений сводится к пониманию процесса как “некоторой совокупности набора исполняющихся команд, ассоциированных с ним ресурсов и текущего момента его

выполнения, находящуюся под управлением операционной системы”.

Конкретизация понятия процесса зависит от целей исследования. Будем рассматривается процесс как последовательность наборов команд (блоков) $I_s = (1,\ 2,\ \ldots,\ s)$. С целью наиболее эффективного решения задач синхронизации процессов, существенной минимизации системных затрат и простоев обрабатывающих устройств, для процессов строятся расписания моментов запуска и окончания каждого из блоков. Моменты времени начала выполнения каждого блока определяются последовательностью $(t_1,\ t_2,\ \ldots,\ t_s)$. Предполагая, что блоки выполняются строго последовательно, в ходе своей реализации являются неделимыми и имеют длительности выполнения $d_j \geq 1$, $j = \overline{1,s}$, получим, что моменты времени завершения выполнения наборов команд определяются последовательностью вида $(t_1 + d_1,\ t_2 + d_2,\ ...,\ t_s + d_s)$.

Для ускорения вычислений процессы могут исполняться псевдопараллельно на одном вычислительном устройстве или параллельно на разных вычислительных устройствах, взаимодействуя между собой. Процессы, которые влияют на поведение друг друга путем обмена информацией, называют

кооперативными или *взаимодействующими* процессами. В связи с этим, *понятие процесса может быть использовано в качестве основного конструктивного элемента для построения параллельных программ в виде совокупности взаимодействующих процессов*.

В ходе своего выполнения состояние процесса может многократно изменяться. Будем считать, что процесс является *активным* и находится в состоянии *выполнения*, если достигается равенство $t_{j+1} = t_j + d_j$, т.е. для выполнения процесса выделено вычислительное устройство, и после завершения выполнения очередного блока процесса сразу же начинается выполнение следующего блока. Соотношение $t_{j+1} > t_j + d_j$ означает, что после выполнения очередного блока процесс *приостановлен* и ожидает возможности для продолжения своего выполнения. Данная приостановка может быть вызвана необходимостью разделения использования единственного вычислительного устройства между одновременно исполняемыми процессами. В этом случае приостановленный процесс находится в состоянии *ожидания* момента предоставления устройства для своего выполнения. Кроме того, приостановка процесса может быть вызвана и временной неготовностью процесса к дальнейшему

выполнению. В подобных ситуациях говорят, что процесс является *блокированным* и находится в состоянии *блокировки*.

Итак, *параллельную программу можно рассматривать как некоторый агрегированный процесс, получаемый путем параллельного объединения составляющих кооперативных (взаимодействующих) процессов*. При разработке таких программ существует ряд особенностей одновременного выполнения нескольких процессов, которые могут быть сформулированы в виде ряда принципиальных положений:

- моменты выполнения командных последовательностей разных процессов могут чередоваться по времени;
- между моментами исполнения блоков разных процессов могут выполняться различные временные соотношения, характер этих соотношений зависит от количества и быстродействия обрабатывающих устройств и загрузки многопроцессорной вычислительной системы и, тем самым, не может быть определен заранее;
- временные соотношения между моментами исполнения блоков могут различаться при разных запусках программ на выполнение, т. е. одной и той же программе при одних и тех же исходных данных могут соответствовать разные последовательности команд

вследствие разных вариантов чередования моментов работы разных вычислительных устройств;

- доказательство правильности получаемых результатов должно проводиться для любых возможных временных соотношений процессов;
- для исключения зависимости результатов выполнения программы от порядка чередования блоков разных процессов необходим анализ ситуаций взаимовлияния процессов и разработка методов для их исключения.

Перечисленные моменты свидетельствуют о существенном повышении сложности параллельного программирования по сравнению с разработкой “традиционных” последовательных программ.

Понятие *ресурса* обычно используется для обозначения любых объектов вычислительной системы, которые могут быть использованы процессом для своего выполнения. В качестве ресурсов могут рассматриваться процессоры, память, программы, данные и т. п. По характеру использования различают следующие категории ресурсов:

- *выделяемые (монопольно используемые)* ресурсы характеризуются тем, что выделяются процессам в момент их возникновения и освобождаются только в

момент завершения процессов (устройства чтения или записи);

- *повторно распределяемые* ресурсы отличаются возможностью динамического запрашивания, выделения и освобождения в ходе выполнения процессов (оперативная память);
- *разделяемые* ресурсы, особенность которых состоит в том, что они постоянно остаются в общем использовании и выделяются процессам для использования в режиме разделения времени (процессор, разделяемые файлы и др.);
- *многократно используемые (реентерабельные)* ресурсы характеризуются возможностью одновременного использования несколькими процессами (реентерабельные программы, файлы, используемые только для чтения и т. д.).

Следует отметить, что тип ресурса определяется не только его конкретными характеристиками, но и зависит от применяемого способа использования. Так, например, оперативная память может рассматриваться как повторно распределяемый, так и разделяемый ресурс, а использование программ может быть организовано в виде ресурса любого рассмотренного типа.

Для параллельного программирования характерной является ситуация, когда одну и ту же программу или ее часть необходимо выполнять многократно, например выполнение циклического участка программы. Многократно выполняемую в многопроцессорной системе программу или ее часть будем называть *программным ресурсом*, а множество соответствующих процессов – *конкурирующими*.

Решая проблему распределения программных ресурсов между процессами, как следствие неявно решаются задачи эффективного использования основных вычислительных ресурсов – процессоров, оперативной памяти, каналов ввода–вывода и др. С этой точки зрения программный ресурс является интегрированным средством по запросам на вычислительные ресурсы. С другой стороны, эффективно решая задачу распределения программных ресурсов, решаем проблему сокращения времени реализации заданных объемов вычислений.

Известно, что производительность многопроцессорных систем (суперкомпьютеров) вычисляться по формуле $P = F \times N \times I$ (операций/секунду).

Здесь

F – тактовая частота процессора в MHz;

N – число процессов (ядер);

I – количество обрабатываемых инструкций (команд) за такт.

Производительность современных супер компьютеров достигает от 10^9 Гигафлопс до 10^{15} Петафлопс.

Итак, с помощью понятий процесса и программного ресурса показано, что создание и исследование программ, использующих принципы параллельной обработки, сводится к решению задач организации взаимодействия параллельных процессов, конкурирующих за программный ресурс.