1.Базовые понятия для заданного алгоритма

1.1)Подсчитываем количество задач для каждой вершины, для которой данная задача будет родительской (вершина, которой соответствует задача, после выполнения которой могут получить данные ближайшие другие задачи(потомки)).

1.2)Сортируем по убыванию время передачи данных от одной задачи к другой и сохраняем в список List\_depend\_tasks.

2.Алгоритм погружения задач на ресурсы системы(процессоры)

2.1)Подгружаем первую попавшуюся независимую задачу на свободный процессор.

2.2)Если задача может выполнится, тоесть не зависит от других задач, то подгружаем её на свободный процессор. При отсутствии свободных процессоров задача ожидает первый освободившейся .

2.3) Синхронно пускаем маркер по логическому кольцу. Переход маркера с процессора на процессор будет занимать 1 такт. Передача данных между процессорами возможна только при наличии маркера у данного процессора.

2.4)Если в результате выполнения задач у нас возникает последующая возможность выполнения зависимых задач от данной, то:

- анализируем родительскую вершину;

-просматриваем список вершин ,которые являются потомками для данной вершины List\_depend\_tasks.

-первую вершину с этого списка подгружаем на тот процессор, на котором находится родительская вершина, что значительно уменьшает время выполнения задания.

2.5)Если после завершения родительской задачи возможно выполнение списка задач-потомков, то мы по определенному алгоритму (алгоритм передачи данных между процессорами) передаем данные для выполнения задач потомков на другие процессоры для эффективной загрузки всей системы.

2.6)Если освободился ресурс, то

если в списке есть задача ,которая готова к выполнения,

то эта задача погружается на данный ресурс. Выбирается со списка та задача, значение весов дуги связи наибольшее.

2.7)Если возникла ситуация когда на одну задачу передаются данные с разных задач, то оставляем на выполнение на своем ресурсе задачу, вес с которой максимальный.

3.Алгоритм передачи данных между процессорами

Разработанный мною алгоритм будет более эффективным и оптимальным при значительной занятости шины, тоесть пересылки между процессорами могут быть больше чем время выполнения задач на ресурсе вычислительной системы.

Данная топология с особенность маркерного доступа обеспечивает эффективную передачу данных между процессорами .Также учитываем продолжительность выполнения каждой задачи.Данная топология позволяет ресурсу одновременно передавать данные и выполнять задачу. Опишем суть алгоритма пошагово:

1)Чтобы предотвратить частое обращение к шине и эффективную передачу между процессорами используется маркер ,который разрешает передачу данных от одного процессора к другому. То есть если процессор требует передачу, то он ожидает маркер.

2)Если процессор захватил маркер, то шина занята и другие процессоры не могу передавать данные. При этом задача на процессоре с маркером, просматривает список задач, которым нужно передать данные.Из списка задач выбирается та задача, у которой вес дуги наименьший ,чтоб уменьшить задержки на пересылки между процессорами.

3)Если возникает ситуация с одинаковыми весами дуг, то выполняется та задача, у которой вес больше. Таким образом мы оптимизируем загрузку ресурсов системы.

4)Пересылка данных осуществляется с помощью буферизации данных. Тоесть все получаемые данные сохраняется в ресурсе системы ,в которой они посылается. Таким образом мы сокращаем задержки на пересылки между процессорами –передаем данные как только их получили.

1)Анализируем весь граф. Находим путь с максимальной длиной пересылки.