Московский государственный университет имени
М.В.Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт

Древовидный маркерный алгоритм

Зыкин Ярослав
428 группа

16 декабря 2020 г.

Содержание

1 Постановка задачи

64 процессов, находящихся на разных ЭВМ сети, одновременно выдали запрос на вход в критическую секцию. Реализовать программу, использующую древовидный маркерный алгоритм для прохождения всеми процессами критических секций.

Критическая секция:

```
1 <проверка наличия файла ‘‘critical.txt’’>;
2 if (<файл ‘‘critical.txt’’ существует>) {
3     <сообщение об ошибке>;
4     <завершение работы программы>;
5 } else {
6     <создание файла ‘‘critical.txt’’>;
7     Sleep (<случайное время>);
8     <уничтожение файла ‘‘critical.txt’’>;
9 }
```

Для передачи маркера использовать средства MPI.

Получить временную оценку работы алгоритма. Оценить сколько времени потребуется, если маркером владеет нулевой процесс. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

2 Описание алгоритма

Алгоритм древовидный маркерный (Raymond)

Все процессы представлены в виде сбалансированного двоичного дерева. Каждый процесс имеет очередь запросов от себя и соседних процессов (1-го, 2-х или 3-х) и указатель в направлении владельца маркера.

Вход в критическую секцию: Если есть маркер, то процесс выполняет КС. Если нет маркера, то процесс:

1. помещает свой запрос в очередь запросов
2. посылает сообщение ЗАПРОС в направлении владельца маркера и ждет сообщений.

Поведение процесса при приеме сообщений: Процесс, не находящийся внутри КС должен реагировать на сообщения двух видов – МАРКЕР и ЗАПРОС.

1. Пришло сообщение МАРКЕР
   (a) Взять 1-ый запрос из очереди и послать маркер его автору (концептуально, возможно себе);
   (b) Поменять значение указателя в сторону маркера;
   (c) Исключить запрос из очереди;
   (d) Если в очереди остались запросы, то послать сообщение ЗАПРОС в сторону маркера.
2. Пришло сообщение ЗАПРОС.
   (a) Поместить запрос в очередь
   (b) Если нет маркера, то послать сообщение ЗАПРОС в сторону маркера, иначе (если есть маркер) – перейти на пункт 1(a).

Выход из критической секции. Если очередь запросов пуста, то при выходе ничего не делается, иначе – перейти к пункту 1(a).

3 Решение

3.1 Идеи, заложенные в решение

- Дерево

  Для работы алгоритма была выбрана стратегия, при которой дерево процессов пронумеровано в ширину. Такая нумерация обусловлена простотой подсчёта номеров вершин-соседей (в этом случае они однозначно определяются номером текущего процесса).

  Всем "закрытым" соседям (родитель для корня, дети для листов) присваивались номера -1

- Маркер

  Маркер как структура данных в программе не реализуется. Показателем того, что маркер находится у текущего процесса, является значение направления к маркеру to_marker, равное своему номеру.

- Запросы

  Запросами реализованы в виде обмена MPI-сообщениями с пустыми телами и типами – типами сообщения.

3.2 Проблемы, возникшие во время решения

- Ошибки компиляции/сборки

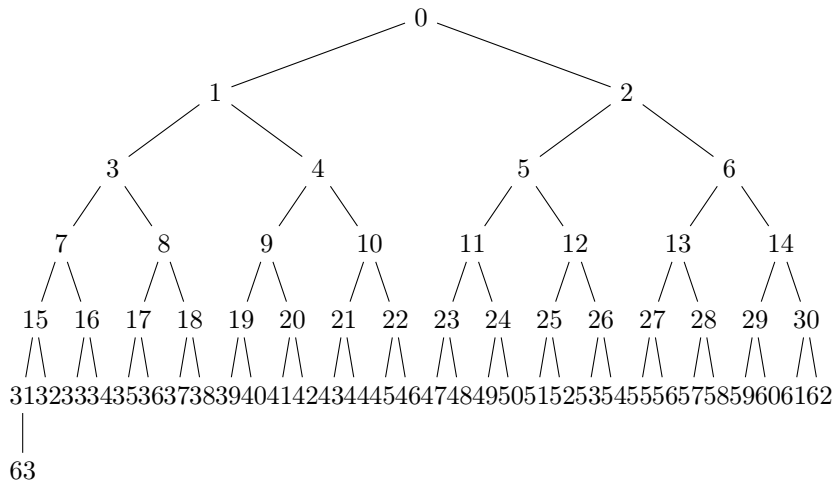
  Стандартная библиотека Си, использованная при компиляции, не содержит функции *rand()* и *sleep()*. Поэтому вместо случайного времени ожидания используется время, равное 1 секунде. Вместо сна используется активное ожидание (*while()* с пустым телом).

- Незавершаемость программы

  Для того, чтобы процесс продолжал передавать запросы и после выхода из критической секции, используется бесконечный цикл *while(1)*. Поэтому программа не завершается самостоятельно (при создании логов для завершении программы посылался сигнал $SIGTERM$).

3.3 Временная оценка

Сбалансированное дерево процессов выглядит так, как показано на рисунке.



В первую очередь, посчитаем, сколько раз маркер будет переходить от одного процесса к другому. Рассмотрим случай оптимального прохождения маркера по дереву – справа налево в глубину. Таким образом передача пройдёт по всем рёбрам, кроме 0-1, 1-3, 3-7, 7-15, 15-31, 31-63, дважды.

$$T_1 = (57 * 2 + 5) * (Ts + 1 * Tb) = 12019$$

Согласно предположению о том, что процессорные операции являются бесконечно быстрыми, будем считать, что критические секции проходятся бесконечно быстро. Чтобы получить итоговую сложность алгоритма, нужно посчитать количество и время для посылаемых и пересылаемых запросов.

1. В худшем случае длина пути, который пройдёт запрос от источника до процесса, содержащего маркер, равна $2 * depth - 1 = 11$. Затраченное время в этом случае: $T_{2_1} = 11 * (Ts + 1 * Tb) = 1111$.

2. Если в момент передачи маркера, процессу-владелецу приходит более одного запроса, то все эти запросы (кроме одного удовлетворённого) будут пересылаться вслед за маркером. В худшем случае наш запрос будет последним в этой очереди, которая с каждой передачей будет уменьшаться на 1. Однако можно считать, что передача всех сообщений одной очереди осуществляется за один сеанс доступа к шине. Получаем в этом случае время:

$$T_{2_2} = \frac{((Ts + num\_proc * Tb) + (Ts + 1 * Tb)) * num\_proc}{2} = 66080$$

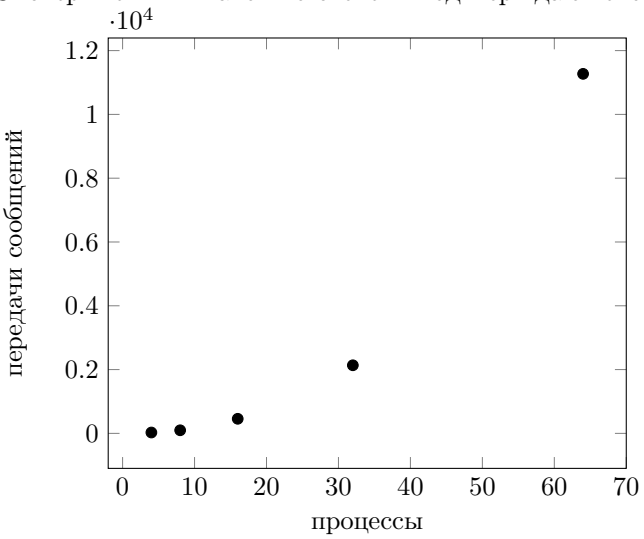
В результате для всего алгоритма имеем:

$$T = T_1 + num\_proc * (T_{2_1} + T_{2_2}) = 4312243$$

3.4 Сравнение с экспериментом

Из подсчёта в предыдущем пункте видно, что сложность данного алгоритма (по количеству передаваемых сообщений) при одновременной попытке всех процессов войти в критическую секцию равна $\mathcal{O}(n^2)$.

Эксперименты в какой-то степени подтверждают этот подсчёт:



3.5 Вывод

Сложность древовидного маркерного алгоритма для запроса на вход в критическую секцию для одного процесса является $\mathcal{O}(\log n)$. Однако если одновременно начнут выдавать запросы большое количество запросов, то сложность резко возрастает. Это связано с тем, что у процессора с маркером формируется большая очередь запросов, которые приходится пересылать вслед за маркером.