

Исследование времени работы
широковещательного алгоритма для
прохождения критических секций.

Санников Тимофей Владимирович

15 октября 2024 г.

Формулировка задачи

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

Для межпроцессорных взаимодействий использовать средства MPI. Получить временную оценку работы алгоритма. Оценить сколько времени потребуется для прохождения всеми критических секций, если маркером владеет нулевой процесс. Время старта равно 100, время передачи байта равно 1 ($T_s = 100$, $T_b = 1$). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

Программная реализация

Для решения данной задачи была реализована программа на языке сrr, соответствующая широковещательному алгоритму прохождения критической секции.

Описание широковещательного алгоритма

- **Предоставление доступа.**

Если маркер у нулевого процесса, он может разрешить процессу i войти. Если нулевой процесс решает, что процесс i может войти в критическую секцию, он передает маркер процессу i .

- **Вход в критическую секцию.**

Процесс i , получив маркер, входит в критическую секцию и выполняет необходимые операции.

Algorithm 1 Критическая секция

```

1: Проверка наличия файла "critical.txt"
2: if файл "critical.txt" существует then
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: end if
```

- **Выход из критической секции.**

После завершения работы в критической секции процесс i выходит из нее и передает маркер обратно нулевому процессу

- **Продолжение работы.**

Нулевой процесс продолжает контролировать доступ к критической секции, принимая запросы от других процессов и управляя передачей маркера.

Описание программной реализации

Из-за равнозначности каждого процесса, претендующего на выполнение критической секции, очередность случайная и не зависит от номера процесса. Для более точного понимания реализации был написан псевдокод 2.

Вычисление времени ожидания каждого процесса:

$$\text{sleep_time} = \text{rand}() \bmod \text{TIME_SLEEP_MAX} + 1 \quad (1)$$

Где TIME_SLEEP_MAX было равно 3, значит каждый процесс ждал от 1 до 3-х секунд.

Оценочное время прохождения критической секции

Для оценки времени работы алгоритма, нужно учитывать следующее:

- **Время старта (T_s):**

100 единиц;

- **Время передачи байта (T_b):**

1 единица;

- **Время выполнения критической секции:**

Это будет зависеть от времени, в течение которого процесс находится в критической секции, которое мы будем считать как T_{cs} , и оно может варьироваться в зависимости от случайного времени, указанного в коде.

Время передачи маркера (T_b): Для передачи маркера от процесса 0 к процессу i потребуется $T_m = T_b = 1$.

Время критической секции (T_{cs}): Пусть T_{cs} - это среднее время, в течение которого процесс находится в критической секции.

Так как T_{cs} у нас варьируется от 1 до 3 секунд (из-за выражения 1), можно рассмотреть его как $T_{cs} = (1 + 2 + 3)/3 = 2$ или 200000 миллисекунд.

Время передачи сообщения о завершении использования критической секции: $T_k = T_b = 1$.

Количество процессов (N): Общее количество процессов в системе. $N = 64$

Общее время для прохождения всех критических секций можно выразить следующим образом:

$$T = T_s + N \cdot (T_m + T_{cs} + T_k) \quad (2)$$

$$T = T_s + N \cdot (T_b \cdot 2 + T_{cs}) \quad (3)$$

T_{cs} — среднее время выполнения критической секции.

Теперь подставим значения в формулу 3:

$$T = 100 + 64 \cdot (1 \cdot 2 + 200000) = 12800228 \quad (4)$$

Таким образом, общее время, необходимое для прохождения всеми процессами критических секций, составляет 12800228 единицу времени.

Итоги

В ходе проведенного исследования была написана программа, имитационно моделирующая широкополосный алгоритм прохождения критической секции, выведена формула для вычисления времени работы данного алгоритма, подставлены значения условия в формулу.

Algorithm 2 Алгоритм критической секции MPI

```
1: Определение rank и size
2: Установка начального значения генератора случайных чисел на основе rank
3: if rank == 0 then
4:   Инициализация remaining_processes как массив от 1 до size-1
5:   while remaining_processes не пустой do
6:     random_index ← случайный индекс от 0 до размера remaining_processes - 1
7:     next_process ← remaining_processes[random_index]
8:     Вывести оставшиеся процессы
9:     Вывести "Процесс 0 отправляет маркер процессу next_process"
10:    Отправить next_process через MPI_Bcast
11:    Ожидать сообщения об освобождении критической секции от next_process
12:    Удалить next_process из remaining_processes
13:   end while
14: else
15:   while истина do
16:     received_marker ← MPI_Bcast
17:     if received_marker == rank then
18:       Войти в критическую секцию
19:       Отправить сообщение процессу 0 об освобождении критической секции через MPI_Send
20:       Выйти из цикла
21:     end if
22:   end while
23: end if
```
