Erm whata sigma

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 ($Ts = 100$, $Tb = 1$). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

Программная реализация

Для решения данной задачи была реализована программа на языке cpp, 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

```
Проверка наличия файла "critical.txt"
if файл "critical.txt"существует then
    Сообщение об ошибке
    Завершение работы программы
else
    Создание файла "critical.txt"
    Sleep(случайное время)
    Уничтожение файла "critical.txt"
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:

$$\text{sleep\_time} = \text{rand}() \mod \text{TIME\_SLEEP\_MAX} + 1 \tag{1}$$

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 (Ts):**
  100 единиц;

- **Время передачи байта (Tb):**
  1 единица;

- **Время выполнения критической секции:**
  Это будет зависеть от времени, в течение которого процесс находится в критической секции, которое мы будем считать как Tcs, и оно может варьироваться в зависимости от случайного времени, указанного в коде.

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

Время критической секции (Tcs): Пусть Tcs - это среднее время, в течение которого процесс находится в критической секции.

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

Время передачи сообщения о завершении использования критической секции: $Tk = Tb = 1$.

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

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

$$T = Ts + N \cdot (Tm + Tcs + Tk) \tag{2}$$

$$T = Ts + N \cdot (Tb \cdot 2 + Tcs) \tag{3}$$

Tcs — среднее время выполнения критической секции.

Теперь подставим значения в формулу 3:

$$T = 100 + 64 \cdot (1 \cdot 2 + 200000) = 12800228 \tag{4}$$

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 широковещательный алгоритм прохождения критической секции, выведена формула для вычисления времени работы данного алгоритма, подставлены значения условия в формулу.

Algorithm 2 Алгоритм критической секции MPI

```
Определение rank и size
Установка начального значения генератора случайных чисел на основе rank
if rank == 0 then
    Инициализация remaining_processes как массив от 1 до size-1
    while remaining_processes не пустой do
        random_index ← случайный индекс от 0 до размера remaining_processes - 1
        next_process ← remaining_processes[random_index]
        Вывести оставшиеся процессы
        Вывести "Процесс 0 отправляет маркер процессу next_process"
        Отправить next_process через MPI_Bcast
        Ожидать сообщения об освобождении критической секции от next_process
        Удалить next_process из remaining_processes
    end while
else
    while истина do
        received_marker ← MPI_Bcast
        if received_marker == rank then
            Войти в критическую секцию
            Отправить сообщение процессу 0 об освобождении критической секции через MPI_Send
            Выйти из цикла
        end if
    end while
end if
```