01-hw-CS

vladrus13rus

September 2021

1 Задача о копировании элементов

Условие: Значение X хранится где-то в памяти EREW PRAM. Покажите, как скопировать X в каждую ячейку массива длины p в EREW PRAM с p процессами за O(log p) time. Определите, за сколько можно сделать тоже самое в CREW и CRCW PRAM.

Поймем, в чем же проблема задачи. Мы находимся в EREW, а это значит, что в один момент времени их ячейки памяти X читает только 1 тредик. Если мы каждый раз будем читать из него, то это займет O(p) времени. Звучит грустно, не так ли? (да, это звучит грустно).

Давайте представим самый оптимальный случай. Мы записываем из N ячеек в N новых ячеек. В следующий шаг запишем $2N$ в $2N$... Лучше мы сделать не сможем, так как умеем читать из одной ячейки в один момент времени только из одного треда (ибо EXCUSIVE READ - ER).

Получим, что у нас на первом моменте времени одна заполненная ячейка (пусть на первом шаге мы сразу запишем в первую ячейку массива), затем 2, затем 4, затем 8, ... Таких шагов (не будем утомлять логарифмами) $log(p)$, то есть это займет такое время (если не считать шага для заполнения первой ячейки).

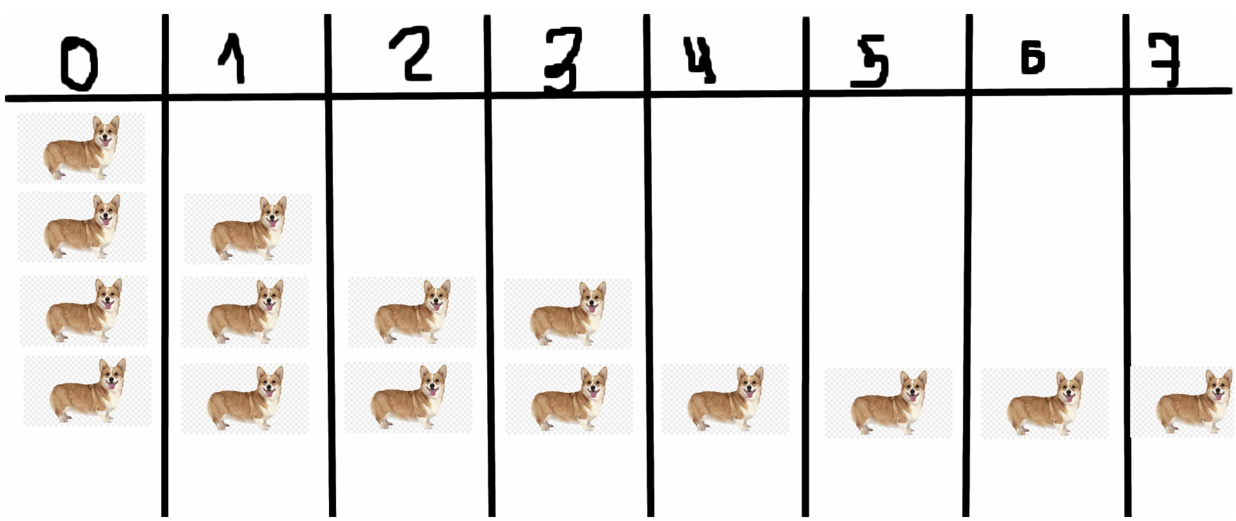

Рисунок 1. Кролики размножаются по Фиббоначии, а корги по степеням двойки

Итог: мы должны на i шаге получить заполненными 2^i элементов массива. Таким образом алгоримт должен работать за O($log(p)$)

```
process(id):
  if (id == 0):
    A[id] <- X
    exit(0) or skip(log(p)) && exit(0)
  for h = 0 to log p:
    if (2^i > id):
      A[id] <- A[id - 2^i]
      exit(0) or skip(log(p) - h) && exit(0)
    else:
      skip(1)
```

Конечно же, здесь предпочтительнее всегда вариант с exit(0), нежели с skip(?) exit(0)

Давайте рассмотрим, как будет себя вести алгоритм при p = 9

id	0	1	2	3	4	5	6	7	8
on step	X	0	1	1	2	2	2	2	3
id from	X	0	0	1	0	1	2	3	0

Как видим, на каждом шаге мы читаем уже записанное раньше значение не более чем в одном треде.

Задача решена!

В случае concurrent read мы можем читать разом все из одной ячейки. Таким образом, каждый тред читает из X и записывает в свою ячейку. Выполняется за O(1)

2 Брент и оптимальный алгоритм

Условие: Докажите, что шедулер из теоремы Брента работает не хуже, чем в 2 раза от оптимального

Какой же алгоритм самый оптимальный? Это тот, который всегда на всех шагах использует все p процессов. То есть он всегда находит работу треду.

Снова рассмотрим улевой строй нашего графа и разобьем наше W на W_0, W_1, ..., W_s. Наш Брент будет простаивать только тогда, когда в слое уже почти закончились задачи и их не хватило на все p тредов. То есть, например, когда у нас 2^k тредов, и на каждом слое с какого то момента $2^i + 1$ задач ($i \geq k$).

Тогда на каждом слое будет простаивать по $p - 1$ треду. Таких слоев может быть до O(S) (пусть они будут от i до j (это упрощенный случай, они могут и как то перемешиваться, но мы берем худший)), и это по настоящему самый сложный случай для нас – мы простаиваем максимальное число тредов максимальное число слоев.

Рассмотрим послойно:

$time_{optimal} = O(1) + W_i/p + W_{i+1}/p + ... + W_j + O(1)$

$time_{Brent} = O(1) + (W_i/p + 1) + (W_{i+1}/p + 1) + ... + (W_j + 1) + O(1) = time_{optimal} + (j - i)$

Заметим, что $j - i$ не больше S (вообще, количество слоев, где у нас в Бренте будет оставаться $p - 1$, конечно же, явно не больше S, так как их общее количество равно S). Но мы точно знаем, что даже самый оптимальный алгоритм не может пройти быстрее чем за

S, как нужно пройти самую длинную цепочку один за другим, потратив как минимум S времени.

$time_{optimal} = X + S$, где $X \geq 0$

$time_{Brent} = time_{optimal} + s$, где $s < S$

$\frac{time_{Brent}}{time_{optimal}} = \frac{X+S+s}{X+S} = 1 + \frac{s}{X+S}$, $s < S$, а значит, что отношение не больше 2, ч.т.д.