Воробйов Виталий

Слуцкий Николай

(Весь этот доклад только для того, чтобы показать что работаем и возможно услышать кое-какие одобрения или замечанию по уже проделанной работе, я имею ввиду, чего добавить, чего удалить. В целом этот доклад, чтобы не занимать время друг друга на разговоры, очень надеюсь что вы прочтёте к какой-то паре, я зайду к Вам и всё выслушаю .\_. Прежде всего оченно Извиняюсь, если что-то в том что я пишу вызовит смущение, недовольство или что-то ещё неприятно. Писалось от души, как говорится.)

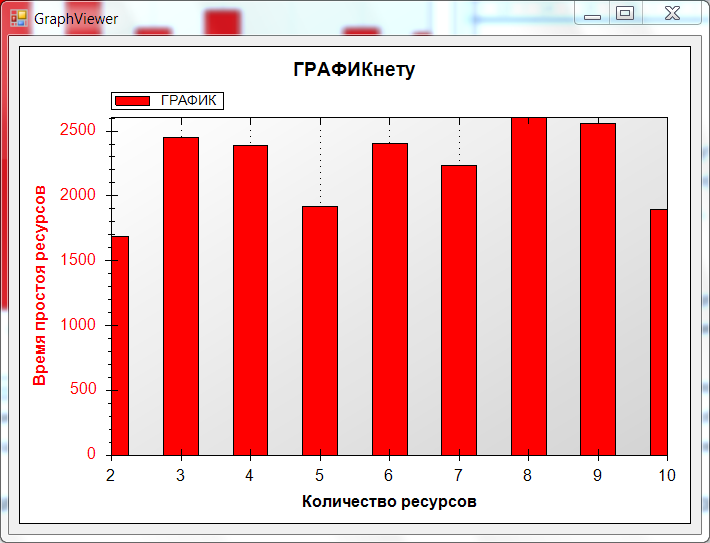
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Потоковая передача (пришла заявка и тут же на ресурс) имеет смысл если заявки идут довольно редко или слишком уж часто (так уж вышло, исходя из моей модели). (по сравнению с тем планированием что я реализовал). При среднем потоке задач, которые успевают к каждой новой сессии собратся в N штук, планирование возможно имеет значение.

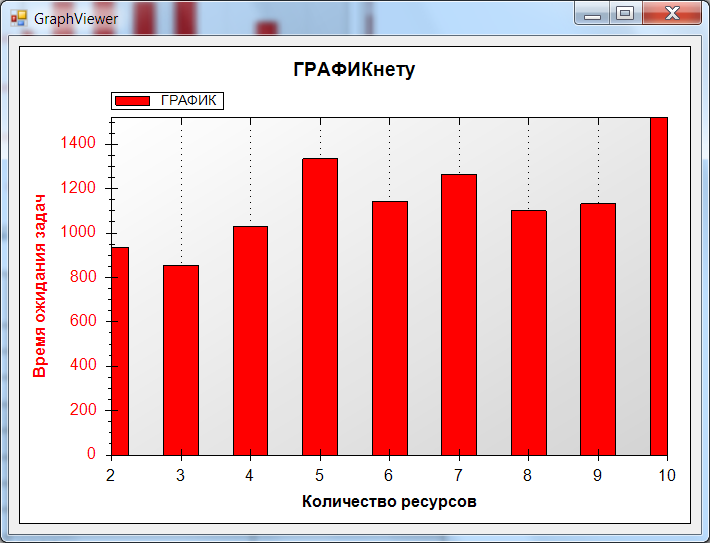
С увеличением ресурсов, увеличивается время ожидания (берётся матрица N\*N) где N - количество ресурсов. Тоесть с увеличением ресурсов увеличивается время ожидания и простоя ресурсов. Особенно это видно на следующих графиках

(в данном случае если время выполнения заказано нормальное, интенсивность появления заявки плавает между 50 и 60 тактами/заявка, вес задачи колеблется от 2 до 35 (чтобы задача успела выполниться к приходу новой заявки) для однородных ресурсов количеством от 2 до 10) ---- (моделировалось 5000 абстрактных тактов GRID'a):

**НЕТ ПЛАНИРОВЩИКА ПЛАНИРОВЩИК СОРТИРОВКИ**

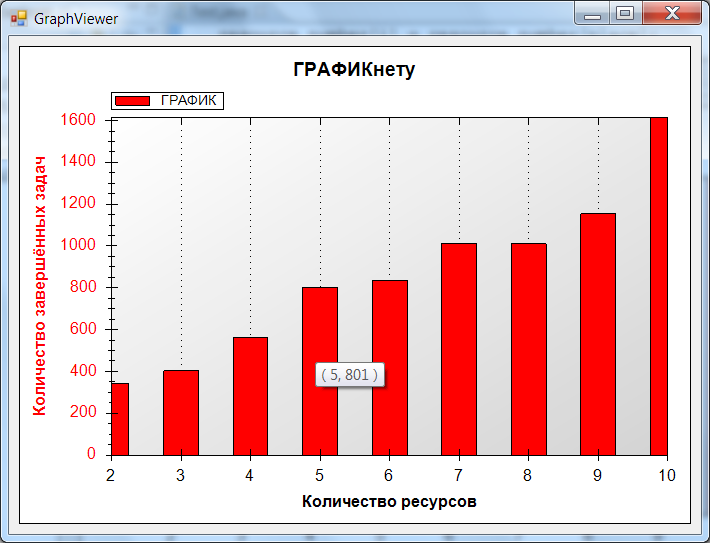
****

|  |  |  |
| --- | --- | --- |
|  | Нет (потоковый режим) | Планирование Симоненко |
| 2 | 1689 | 2501 |
| 3 | 2451 | 2969 |
| 4 | 2385 | 3079 |
| 5 | 1919 | 3243 |
| 6 | 2406 | 2700 |
| 7 | 2236 | 2876 |
| 8 | 2602 | 2341 |
| 9 | 2562 | 2702 |
| 10 | 1895 | 2654 |



|  |  |  |
| --- | --- | --- |
|  | Нет (потоковый режим) | Планирование Симоненко |
| 2 | 934 | 1594 |
| 3 | 853 | 1898 |
| 4 | 1030 | 1948 |
| 5 | 1334 | 1902 |
| 6 | 1143 | 1998 |
| 7 | 1266 | 1388 |
| 8 | 1101 | 1680 |
| 9 | 1131 | 1281 |
| 10 | 1519 | 1111 |

И естественно меньше выполненных задач (упала производительность системы, хм, плохо)



|  |  |  |
| --- | --- | --- |
|  | Нет (потоковый режим) | Планирование Симоненко |
| 2 | 344 | 261 |
| 3 | 402 | 318 |
| 4 | 561 | 386 |
| 5 | 801 | 455 |
| 6 | 835 | 730 |
| 7 | 1014 | 766 |
| 8 | 1010 | 1133 |
| 9 | 1115 | 1076 |
| 10 | 1613 | 1246 |

НО С УВЕЛИЧЕНИЕ КОЛИЧЕСТВА УЗЛОВ, может быть улучшение выполнения. В целом, если задачи неоднородные и узлы тоже весьма неоднородные, то для некоторых задач необходимы быстрые узлы. В режиме потока эти задачи выполнятся быстрее, если выполнять такое планирование, если есть пул задач требующих несколько ресурсов, несколько из которых указаны в требуемости обеих задач.

Возможно также, что так как эти заявки преимущественно маленькие, то потому быстро делаются, или потому что тот момент который надо будет промоделировать (он указан в красных строках под номером 2) практически не попадается.

Фишка такого плохого результата в том, что всё на всете ждёт много времени, пока заполнится буффер на N задач (для N ресурсов), и тогда только потом идёт перепланировывание. Нужно переходить от задачи паросочетания к NP-полному алгоритму пространственно-временного планирования, мне так кажется.

**1!!!!!!**Возможно стоит ещё поработать над учётом неоднородности ресурсов.. тоесть для больших задач брать исключительно те ресурсы которые быстро выполняют, тоесть максимально свести какбы время выполнения задач на всех ресурсах к одному времени, к следующему раз (на этой неделе) запилим и принесём вместе с курсовой, если выдадите разрешение на печать... Если нет... Придётся красить волосы в фиолетовый или синий цвет :)

**2!!!!!!**Ещё надо будет промоделировать момент когда заявка ждёт в очереди освобождения ресурса (если нет планировщика) и посмотреть назначится ли ей ресурс, если производить планирование. Я чего-то вцепился в эту "модель" и совсем забыл про основное задание, к следующему разу исправлюсь и принесу ещё и эти результаты моделирования, но они будут результатами другой программы, более строгой и менее гибкой. Тоесть я продумаю несколько ситуаций и смоделирую.

1. АЛГОРИТМ ПЛАНИРОВАНИЯ

//1. Находим сумму элементов в строке

//2. Находим строку с мин. Суммой элементов и переставляем эту строку, со строкой, соотв. Номеру итерации

//3. Находим сумму элементов в столбцах.

//4. Находим столбец, который содержит в строке, соотв. Номеру итерации 1 и имеет макс. Сумму

//5. Переставляем этот столбец со столбцом, соотв. Номеру итерации

//6. Удаляем первый столб. И первую строку

//7. Переходим к след. Итерации с 1 пункта

//8. После сортировки выбираем ресурс для погружения задачи следующим образом:

//8.1. Так как отсортированные находятся ближе всего к левому краю, идём справа, если встречаем единичку, на этом ресурсе, идём дальше, иначе обозначаем этот ресурс 2. (пересечение задания и ресурса).

//8.2. Обход идти сверху, потому что сверху идуть САМЫЕ требовательные задачи.

Пример работы:

1 2 3 4 5 6 7 8

1 0 0 0 1 1 0 1 : 1

0 0 0 1 1 0 1 1 : 2

1 1 1 0 1 1 0 1 : 3

1 0 0 0 0 0 0 0 : 4

0 1 1 0 0 0 0 0 : 5

1 0 1 0 1 0 0 1 : 6

0 0 1 0 0 0 0 1 : 7

0 1 0 0 1 0 1 0 : 8

5 2 3 4 1 6 7 8

0 0 0 0 1 0 0 0 : 4

1 0 0 1 0 0 1 1 : 2

1 1 1 0 1 1 0 1 : 3

1 0 0 0 1 1 0 1 : 1

0 1 1 0 0 0 0 0 : 5

1 0 1 0 1 0 0 1 : 6

0 0 1 0 0 0 0 1 : 7

1 1 0 0 0 0 1 0 : 8

5 8 3 4 1 6 7 2

0 0 0 0 1 0 0 0 : 4

0 0 1 0 0 0 0 1 : 5

1 1 1 0 1 1 0 1 : 3

1 1 0 0 1 1 0 0 : 1

1 1 0 1 0 0 1 0 : 2

1 1 1 0 1 0 0 0 : 6

0 1 1 0 0 0 0 0 : 7

1 0 0 0 0 0 1 1 : 8

5 8 3 4 1 6 7 2

0 0 0 0 1 0 0 0 : 4

0 0 1 0 0 0 0 1 : 5

0 1 1 0 0 0 0 0 : 7

1 1 0 0 1 1 0 0 : 1

1 1 0 1 0 0 1 0 : 2

1 1 1 0 1 0 0 0 : 6

1 1 1 0 1 1 0 1 : 3

1 0 0 0 0 0 1 1 : 8

5 8 3 1 4 6 7 2

0 0 0 1 0 0 0 0 : 4

0 0 1 0 0 0 0 1 : 5

0 1 1 0 0 0 0 0 : 7

1 1 1 1 0 0 0 0 : 6

1 1 0 0 1 0 1 0 : 2

1 1 0 1 0 1 0 0 : 1

1 1 1 1 0 1 0 1 : 3

1 0 0 0 0 0 1 1 : 8

5 8 3 1 6 4 7 2

0 0 0 1 0 0 0 0 : 4

0 0 1 0 0 0 0 1 : 5

0 1 1 0 0 0 0 0 : 7

1 1 1 1 0 0 0 0 : 6

1 1 0 1 1 0 0 0 : 1

1 1 0 0 0 1 1 0 : 2

1 1 1 1 1 0 0 1 : 3

1 0 0 0 0 0 1 1 : 8

5 8 3 1 6 7 4 2

0 0 0 1 0 0 0 0 : 4

0 0 1 0 0 0 0 1 : 5

0 1 1 0 0 0 0 0 : 7

1 1 1 1 0 0 0 0 : 6

1 1 0 1 1 0 0 0 : 1

1 1 1 1 1 0 0 1 : 3

1 1 0 0 0 1 1 0 : 2

1 0 0 0 0 1 0 1 : 8

5 8 3 1 6 7 4 2

0 0 0 1 0 0 0 0 : 4

0 0 1 0 0 0 0 1 : 5

0 1 1 0 0 0 0 0 : 7

1 1 1 1 0 0 0 0 : 6

1 1 0 1 1 0 0 0 : 1

1 1 1 1 1 0 0 1 : 3

1 1 0 0 0 1 1 0 : 2

1 0 0 0 0 1 0 1 : 8

5 8 3 1 6 7 4 2

0 0 0 2 0 0 0 0 : 4

0 0 1 0 0 0 0 2 : 5

0 1 2 0 0 0 0 0 : 7

1 2 1 1 0 0 0 0 : 6

1 1 0 1 2 0 0 0 : 1

2 1 1 1 1 0 0 1 : 3

1 1 0 0 0 1 2 0 : 2

1 0 0 0 0 2 0 1 : 8

В результате погружаем 1 задачу на 6 ресурс

2 задачу на 4 ресурс

3 задачу на 5 ресурс

4 задачу на 1 ресурс

5 задачу на 2 ресурс

6 задачу на 8 ресурс

7 задачу на 3 ресурс

8 задачу на 7 ресурс

**ДРУГИЕ ВАРИАНТЫ ПЛАНИРОВАНИЯ**

Так как времени на курсовую было мало, всего 4 дня (если будние не учитывать), был реализован минимум, если бы было время без возможности потери оценки... :3

**1.** Как вариант есть предложение реализовать алгоритм пространственно-временной (если правильно понимаю, это когда задач меньше чем ресурсов?), тоесть брать например матрицу до 5 задач на большее количество ресурсов, но тогда надо будет подумать над видоизменением алгоритма или если не надо видоизменять, то просто подумать.

**2.** Можно как то определять частоту появления новых заявок и простаивающиеся ресурсы (пока наберётся достаточно задач для планирования) заполнять лёгкими задачами, которые могут занять ресурс и выйти раньше чем наконец соберётся пачка заявок для сортировки и назначения матрицы N\*N. Хотя мне почемуто кажется что момент с определением частоты слишком притянут за ушы и/или сложный и/или бредовый, но в приципе это вполне адаптивно.

**3.** Развитие идеи первого (**1**) варианта... МОжно потыкать посмотреть, если брать ровно половину (когда колво задач равно ровно половину ресурсов), чё и как будет. Можно брать константное число... только 5 задач и сразу планирвание независимо от количества ресурсов (если их конечно не меньше 6)! Что-то можно придумать и подумать както...

**4.** Тоже бредовая идея, можно ввести приоритеты... 1 и 0. Если заявка высокого приоритета, то мы можем определять максимальный по полученным требованиям (указанные требования могут быть не только высокая performance ресурса) для задачи ресурс (свободный естественно) и сразу в потоке ему кидать. Если заявка не срочная, можно конечно обеспечить планирование (ведь для чего оно вообще... чтобы разбросать заявки грамотно и чтобы все задачи более менее выполнились, тогда как в потоковом всё на свете позанимали и получилось что заявка новая ждёт и простаивает долго в очереди, хотя при планировании всё ок, всё чётко... хм).

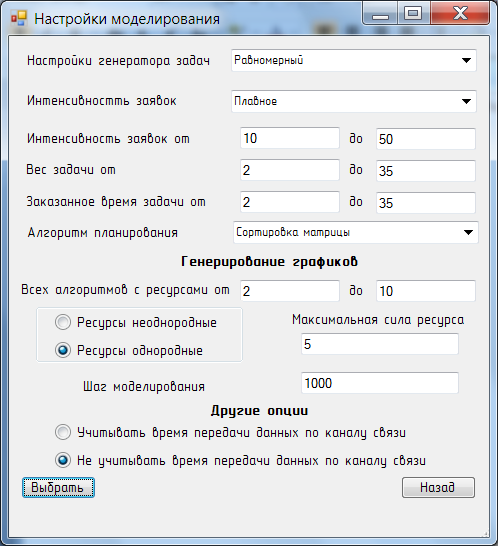
Если за эту ночь Воскресенье-Понедельник ещё чтото в голову придёт я напишу уже ручками ниже, так что извините за столь корявый почерк.

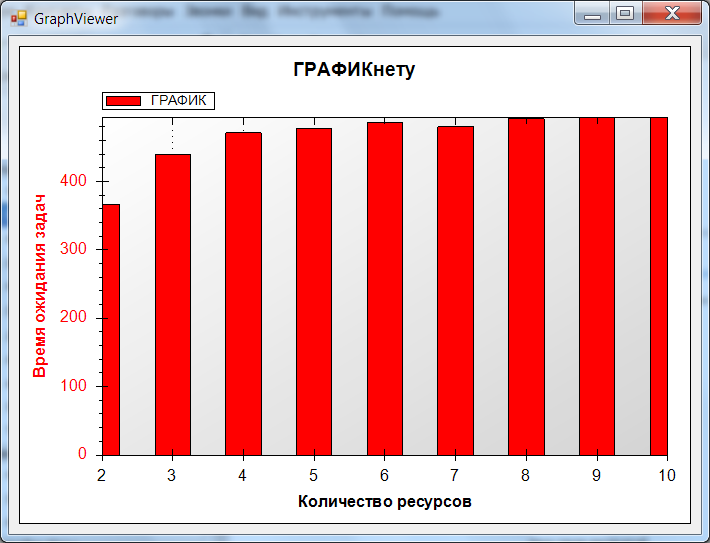
**5.**

**6.**

**Ну и ещё коекакие результаты моделирования**

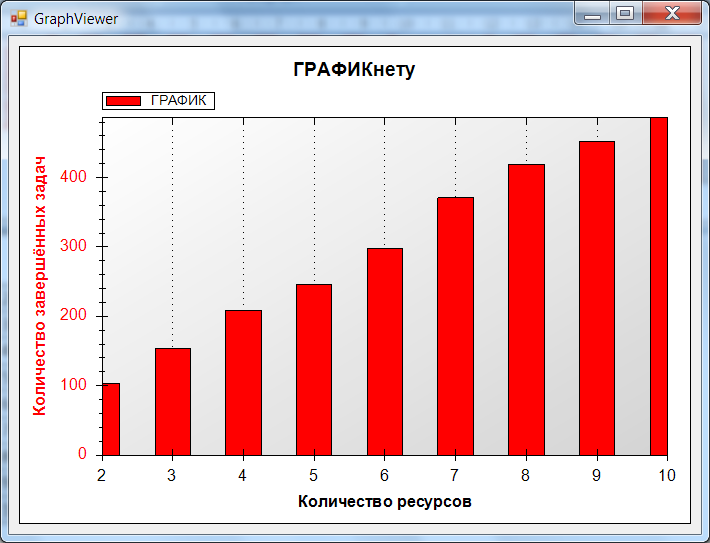
**ВХОДНЫЕ ДАННЫЕ:**

****



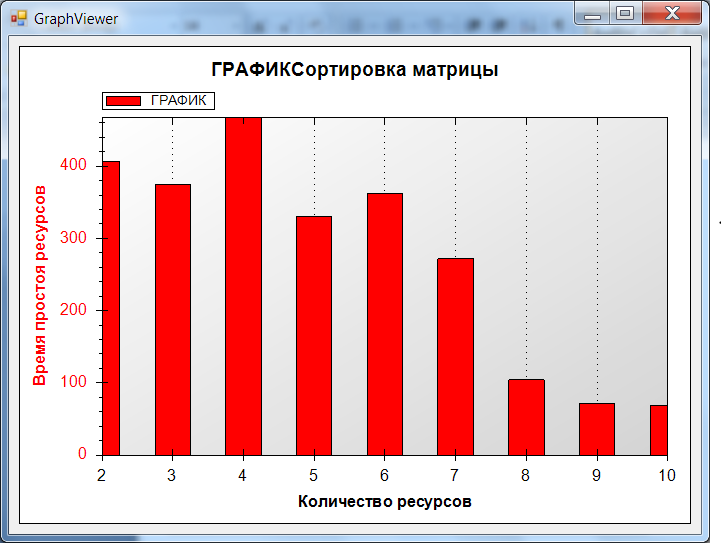
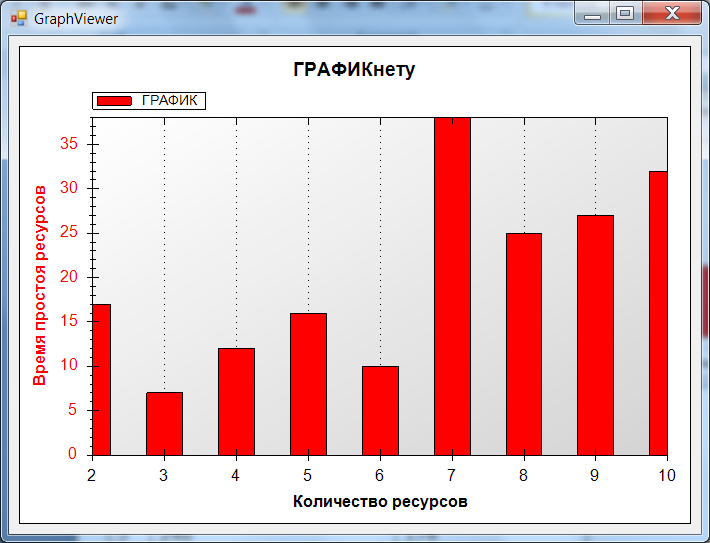
|  |  |  |
| --- | --- | --- |
|  | Нет (потоковый режим) | Планирование Симоненко |
| 2 | 366 | 702 |
| 3 | 439 | 755 |
| 4 | 471 | 801 |
| 5 | 478 | 786 |
| 6 | 486 | 803 |
| 7 | 480 | 720 |
| 8 | 492 | 599 |
| 9 | 493 | 561 |
| 10 | 493 | 476 |

Вот кстати интересно, с такими входными данными с увеличением количества ресурсов алгоритм планирования начинает показывать улучшения. Возможно тут попался тот случай который надо будет отдельно моделировать (2 строка красным написанное выше).



|  |  |  |
| --- | --- | --- |
|  | Нет (потоковый режим) | Планирование Симоненко |
| 2 | 103 | 63 |
| 3 | 154 | 103 |
| 4 | 209 | 111 |
| 5 | 246 | 178 |
| 6 | 298 | 190 |
| 7 | 371 | 258 |
| 8 | 419 | 378 |
| 9 | 452 | 424 |
| 10 | 486 | 497 |

И даже задач больше вышло из системы... хм...



|  |  |  |
| --- | --- | --- |
| P | Нет (потоковый режим) | Планирование Симоненко |
| 2 | 17 | 406 |
| 3 | 7 | 374 |
| 4 | 12 | 467 |
| 5 | 16 | 331 |
| 6 | 10 | 362 |
| 7 | 38 | 272 |
| 8 | 25 | 104 |
| 9 | 27 | 72 |
| 10 | 32 | 68 |

НО ресурсы простаивают конечно всё равно...