Задача построения расписаний

Шпигун Игорь Кириллович

4 ноября 2024 г.

1 Формальная постановка задачи

1.1 ДАНО

- M – количество процессоров, $\{P_1, \ldots, P_M\}$ – множество процессоров
- $N = (Task_1, Task_2, \ldots, Task_N)$ – множество независимых работ
- $T = (Time_1, Time_2, \ldots, Time_N)$ – время выполнения работ, где $Time_i$ время выполнения работы $Task_i$, $Time_i > 0$

  Определим расписание S как множество $\{S_i\}$, где $S_i = (P_j, Time_i^{start}, Time_i^{end})$, где P_j — процессор, на котором выполняется работа $Task_i$, $Time_i^{start}$ — время начала работы $Task_i$, $Time_i^{end} = Time_i^{start} + W_i$ — время завершения работы $Task_i$.

1.2 ТРЕБУЕТСЯ

Построить расписание выполнения всех N работ на M процессорах без прерываний, чтобы минимизировать критерий указанный далее .

Необходимо привязать каждую работу $Task_i$ к некоторому процессору j, на котором будет выполняться данная работа, задать порядок выполнения работ на каждом из процессоров.

1.3 МИНИМИЗИРУЕМЫЙ КРИТЕРИЙ

Разбалансированность расписания

$$K = T_{\max} - T_{\min},$$

где:

$$T_{\max} = \max_{j=1,2,\ldots,M} T_j,$$

$$T_{\min} = \min_{j=1,2,\ldots,M} T_j,$$

$T_j = \sum_{i \in \mathcal{J}_j} Time_i$ — время выполнения всех назначенных на процессор j работ, $\mathcal{J}_j$ — множество всех работ определенных на выполнение на данном процессоре, причем, учитывая что все работы будут выполнены, $\sum_{j=1}^{M} \mathcal{J}_j = N$.

2 Ограничения на корректность расписания

- $\forall i \in [1,\ N]\ \exists! j \in [1,\ M]$: работа $Task_i$ привязана к процессору с номером j
- $Time_i^{end} = Time_k^{start}$ где $S_i = (P_j, Time_i^{start}, Time_i^{end})$, $S_k = (P_j, Time_k^{start}, Time_k^{end})$ и работа k начинает выполняться сразу после работы i. То есть, иными словами, время между концом одной работы и началом следующей работы на одном и том же процессоре равно 0.
- $\forall i \in [1,\ N]\ \ Time_i^{end}$ - $Time_i^{start} = const_i$. Время выполнения каждой работы $Time_i$ фиксировано

3 Рассматриваемые законы изменения температуры

- Cauchy: $T = T_0 \frac{1}{1+i}$
- Log: $T = T_0 \frac{log(1+i)}{1+i}$
- Boltzmann: $T = T_0 \frac{1}{log(1+i)}$

где i – номер итерации.

4 Исследование последовательной реализации

4.1 Экспериментальное исследование

Для Исследования работы последовательного алгоритма был проведен запуск программы, получающей на вход различное количество задач, число которых росло пока время распределения по процессорам не достигло одной минуты. Задачи имели случайно сгенерированное время выполнения от 1 до 100. Полученные наборы задач были распределены с использованием различных законов понижения температуры.

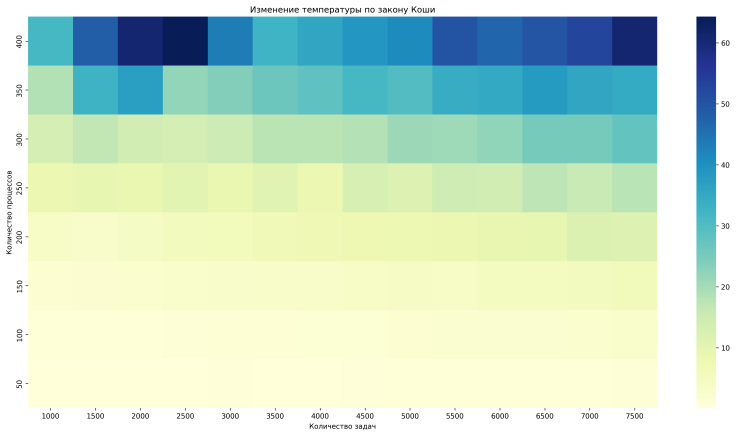

Рис. 1: По закону Коши

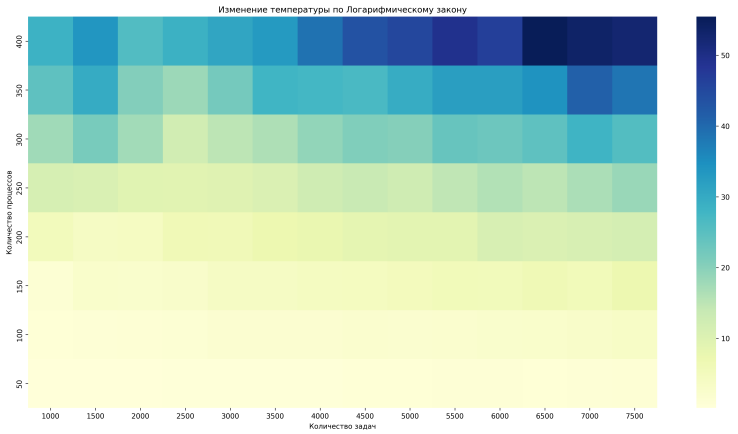

Рис. 2: По Логарифмическому закону

Исходя из полученных графиков можно сделать вывод, что время выполнения распределения растет при росте количества процессоров и количества задач. Дольше всего работает закон Больцмана и при этом показывает не самое лучшее значение критерия. Быстрее всех работает Логарифмический закон и имеет точность, схожую с законом Коши.

4.2 Исследование параллельной реализации

Схема работы параллельного алгоритма имеет практически следующий вид(рис. 3), но только у всех потоков будет одинаковое начальное решение.

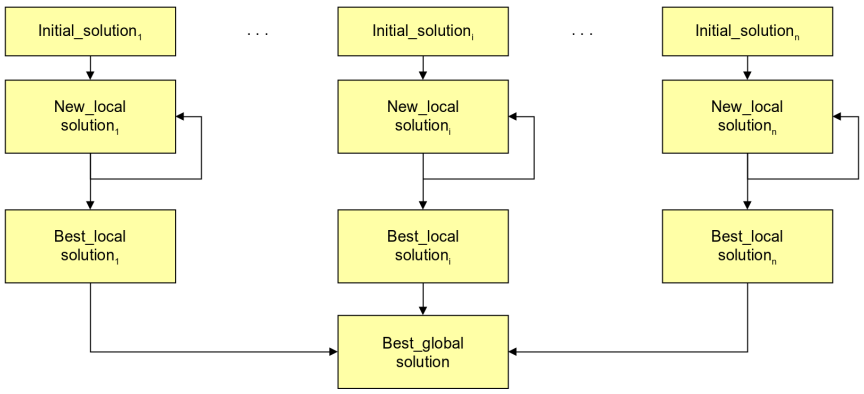

Рис. 3: Параллельная реализация

Была проведена серия экспериментов для двух типов задач (100 задач и 20 процессов , 7500 задач и 400 процессов). В первом случае было проведено по 100 запусков программы, во втором по 10, так как каждый запуск работал около 1 минуты. Реализация получилась такой, что нет никакого выигрыша по времени программы, при любом количестве потоков программа будет работать приблизительно одинаково, при этом явно видно, что при росте количества потоков качество становится лучше. В связи с этим предлагается использовать именно параллельную реализацию, а не последовательную.

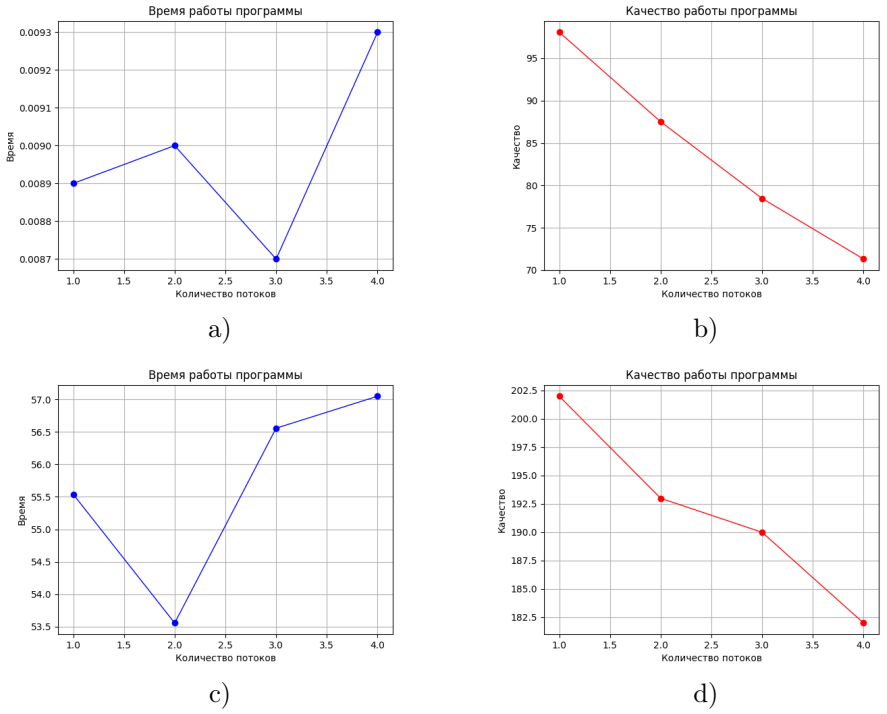

Рис. 4: Результаты работы параллельного алгоритма (время и качество) при 100 задачах и 20 проыцессах (a и b) и при 7500 задачах и 400 процессах (c и d)