Проектная работа.
Knapsack problem

Студент 01: Статный Дмитрий
Студент 02: Смирнов Алексей

Весенний семестр

Overview

1. Одномерный рюкзак
2. Многомерный рюкзак
3. Одномерный Мультирюкзак
4. Многомерный Мультирюкзак
5. Дальнейшие планы
6. [Доп.] SCSK. Построение матрицы Q
7. [Доп.] MCSKS. Построение матрицы Q
8. [Доп.] MCMKS. Построение матрицы Q

Одномерный рюкзак

Формулировка проблемы

Дано N предметов, n_i предмет имеет массу $w_i > 0$ и стоимость $c_i > 0$. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

Одномерный рюкзак

Формулировка проблемы

Дано N предметов, n_i предмет имеет массу $w_i > 0$ и стоимость $c_i > 0$. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

Линейно-алгебраическая формулировка

Дано два вектора $w = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n$ и $c = (c_1, c_2, \ldots, c_n) \in \mathbb{R}^n$. Нужно построить битовый вектор $b \in \{0, 1\}^n$:

$$\begin{cases} (w, c) \to \mathbf{max} \\ (w, b) \leqslant W \end{cases}$$

Не будем задерживаться на простой постановке задачи и перейдём сразу к многомерному случаю.

Многомерный рюкзак

Формулировка проблемы в терминах задач и серверов

Дано N задач, i задача имеет следующие параметры: потребление памяти диска $\alpha_i > 0$, потребление оперативной памяти $\beta_i > 0$ и требует ядер $\gamma_i > 0$. Необходимо выбрать такое подмножество задач, чтобы их суммарное потребление памяти на диске не превосходило A, количество ядер – B, а потребление оперативной памяти – C, но при этом распределение ресурсов было максимально.

Описание тестов

Тестирование будем производить на Simulated Annealing, который основывается на имитации физического процесса, который происходит при кристаллизации вещества.

Описание тестов

Тестирование будем производить на Simulated Annealing, который основывается на имитации физического процесса, который происходит при кристаллизации вещества.

Рассмотрим набор из 500 задач.

Тест №1. num_reads: 1000

Видно, что заполнение происходит на довольно хорошем уровне.

Видно, что заполнение происходит на довольно хорошем уровне.

А теперь рассмотрим больший масштаб: с набор из 5000 задач!
Соответственно, увеличим и ограничения.

Выводы

Одномерный Мультирюкзак

Формулировка в терминах предметов и рюкзаков

Дано N предметов, M рюкзаков, n_i предмет имеет массу $w_i > 0$ и стоимость $v_i > 0$. Необходимо распределить из этих предметов такое подмножество предметов по M рюкзакам, чтобы масса предметов в i-м рюкзаке не превосходила заданной величины c_i (вместимость i-го рюкзака), а суммарная стоимость была максимальна.

Многомерный Мультирюкзак

Формулировка в терминах задач и серверов

Дано N задач, M серверов, i задача имеет следующие параметры: потребление памяти диска $\alpha_j > 0$, потребление оперативной памяти $\beta_j > 0$ и требует ядер $\gamma_j > 0$. Необходимо найти такое подмножество задач, чтобы распределение ресурсов по M серверам было максимально, но количество ядер потребляемых совокупностью задач на i сервере не превосходило заданной величины A_i (вместимость в ядрах i-го сервера), по потребляемой памяти диска – B_i (вместимость i-го сервера по памяти диска), а также по оперативной памяти – C_i. Более того, каждая задача может быть запущена не более чем K раз (если не оговорено иного, $K = 1$).

Описание тестов

Рассмотрим постановку, в которой есть 12 серверов. Сначала для набора из 300 задач, а затем – из 2000.

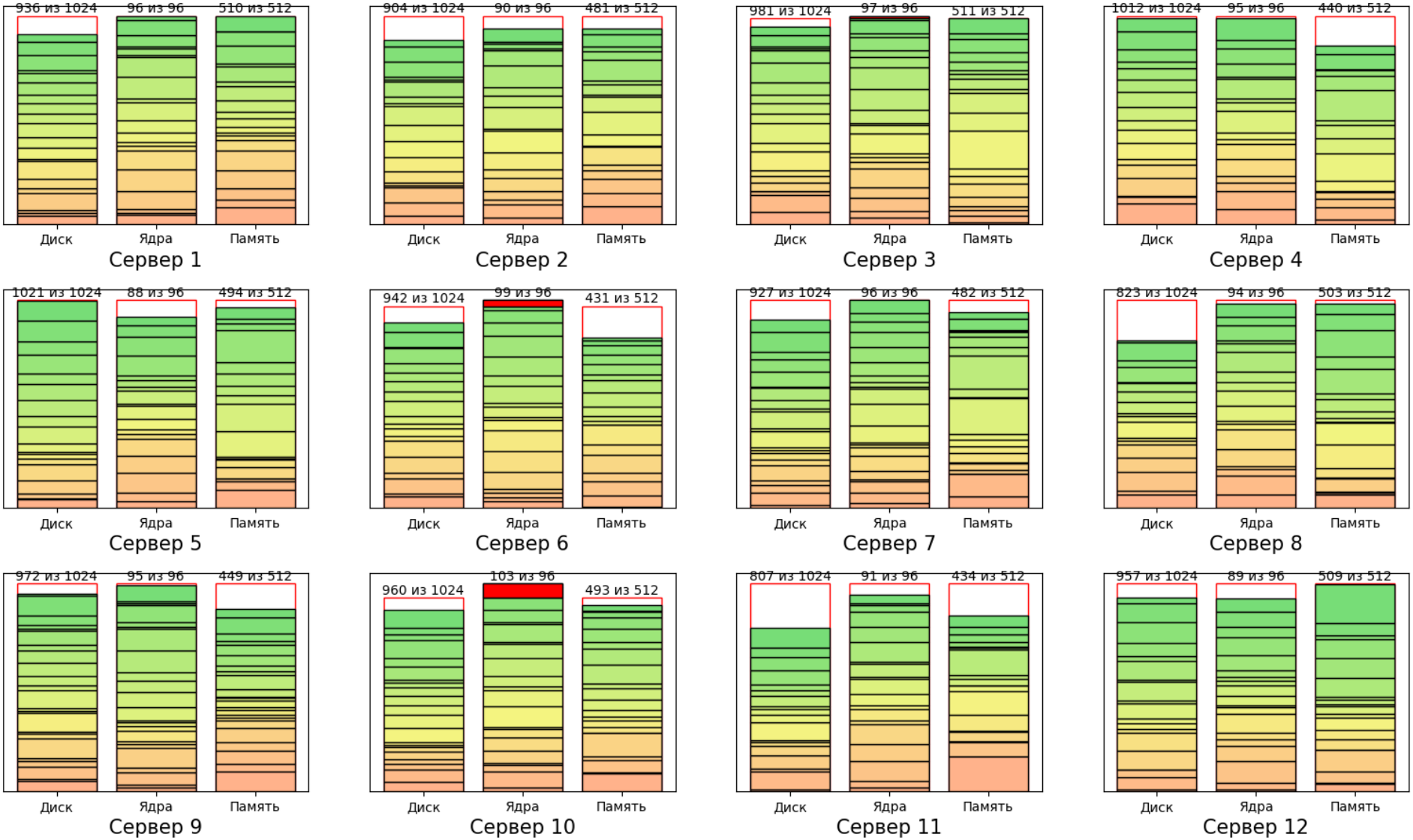
Описание тестов

Рассмотрим постановку, в которой есть 12 серверов. Сначала для набора из 300 задач, а затем – из 2000.

Настройки при тестировании и `sampler` возьмём с многомерного рюкзака.

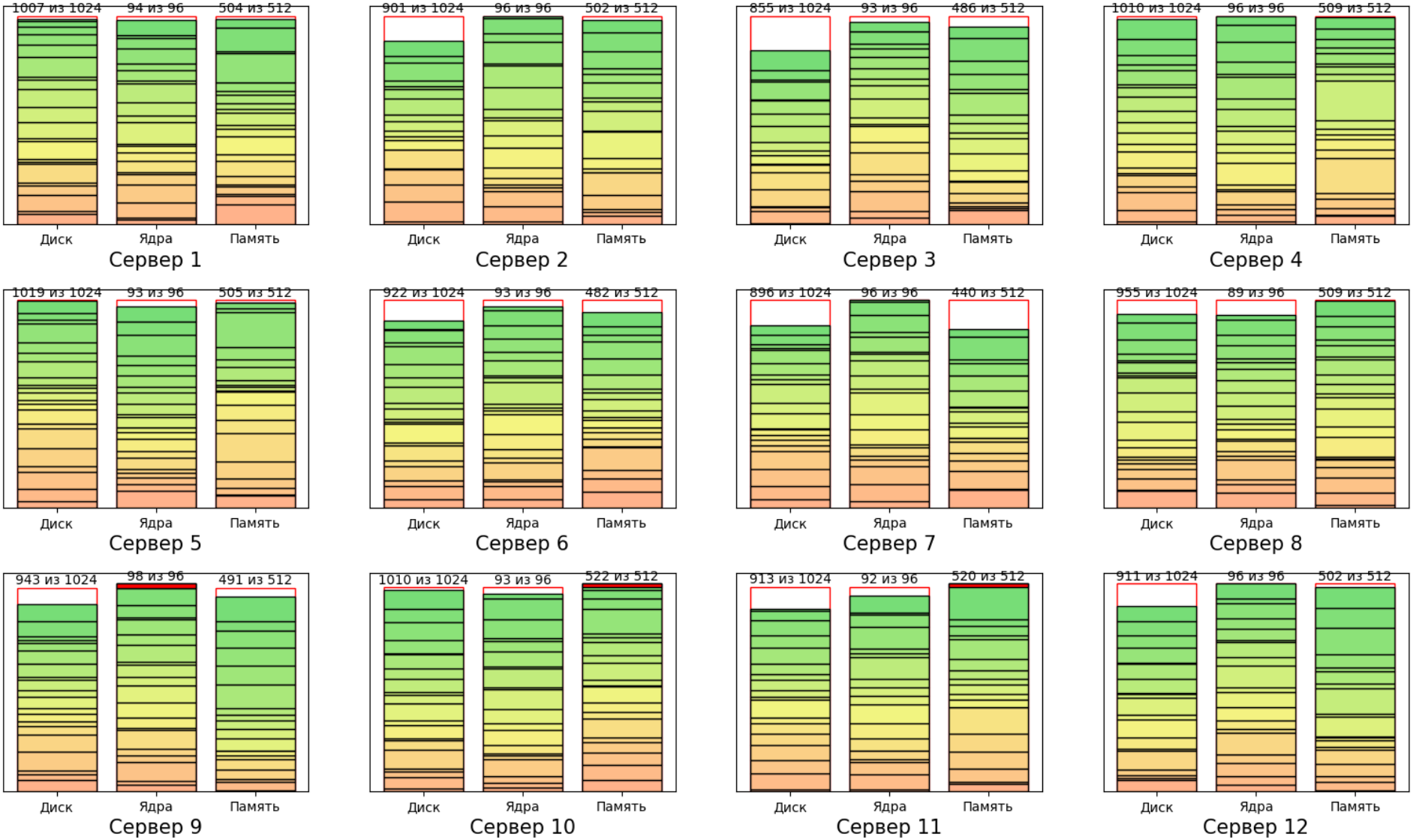
Тест №1 (1). num_reads: 10

Распределение ресурсов на сервере. Количество задач: 300. Значение num_reads: 10. Время работы: 7.037 s.



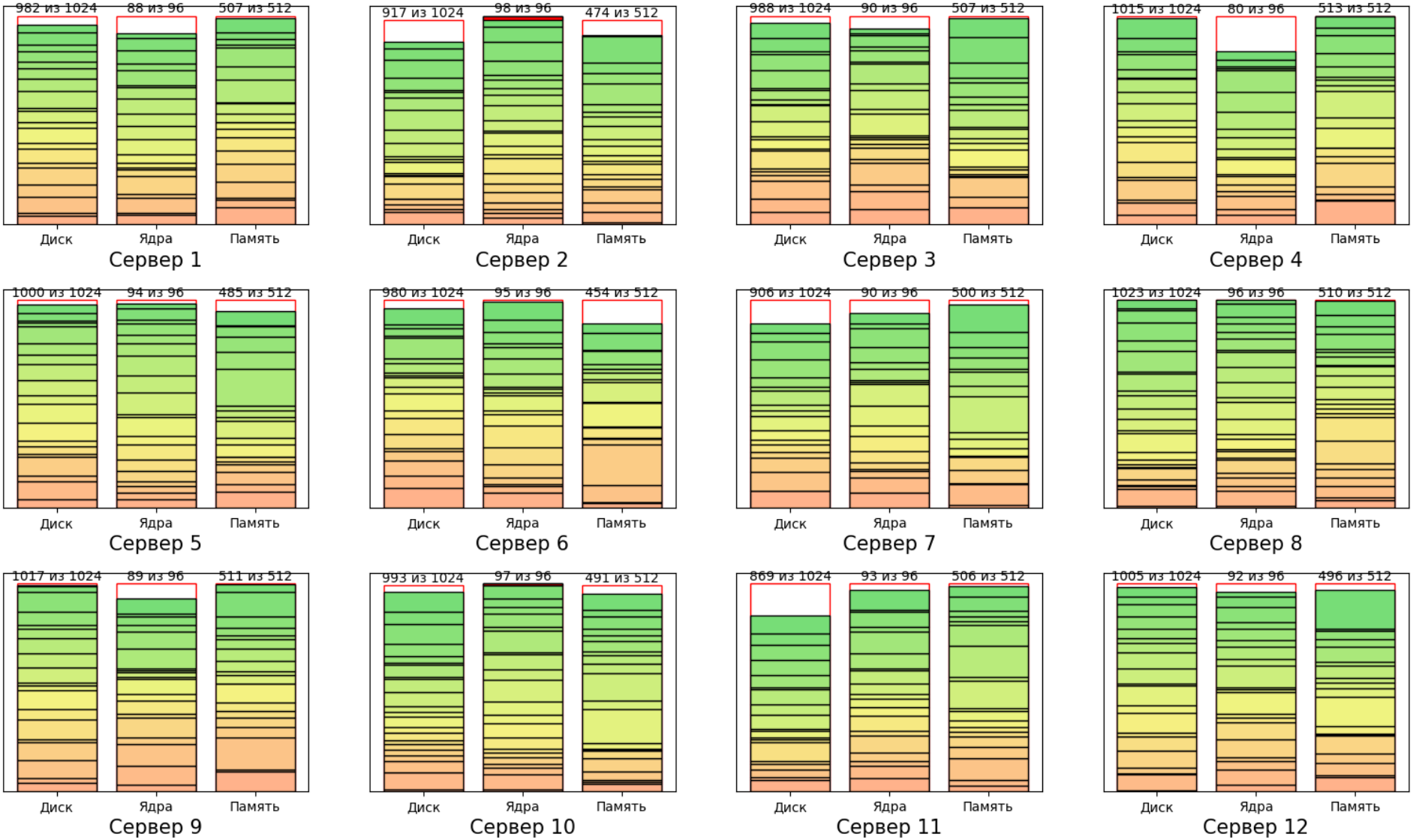
Тест №1 (2). Увеличиваем num_reads до 100

Распределение ресурсов на сервере. Количество задач: 300. Значение num_reads: 100. Время работы: 70.307 s.



Тест №1 (3). Увеличиваем num_reads до 500

Распределение ресурсов на сервере. Количество задач: 300. Значение num_reads: 500. Время работы: 307.003 s.



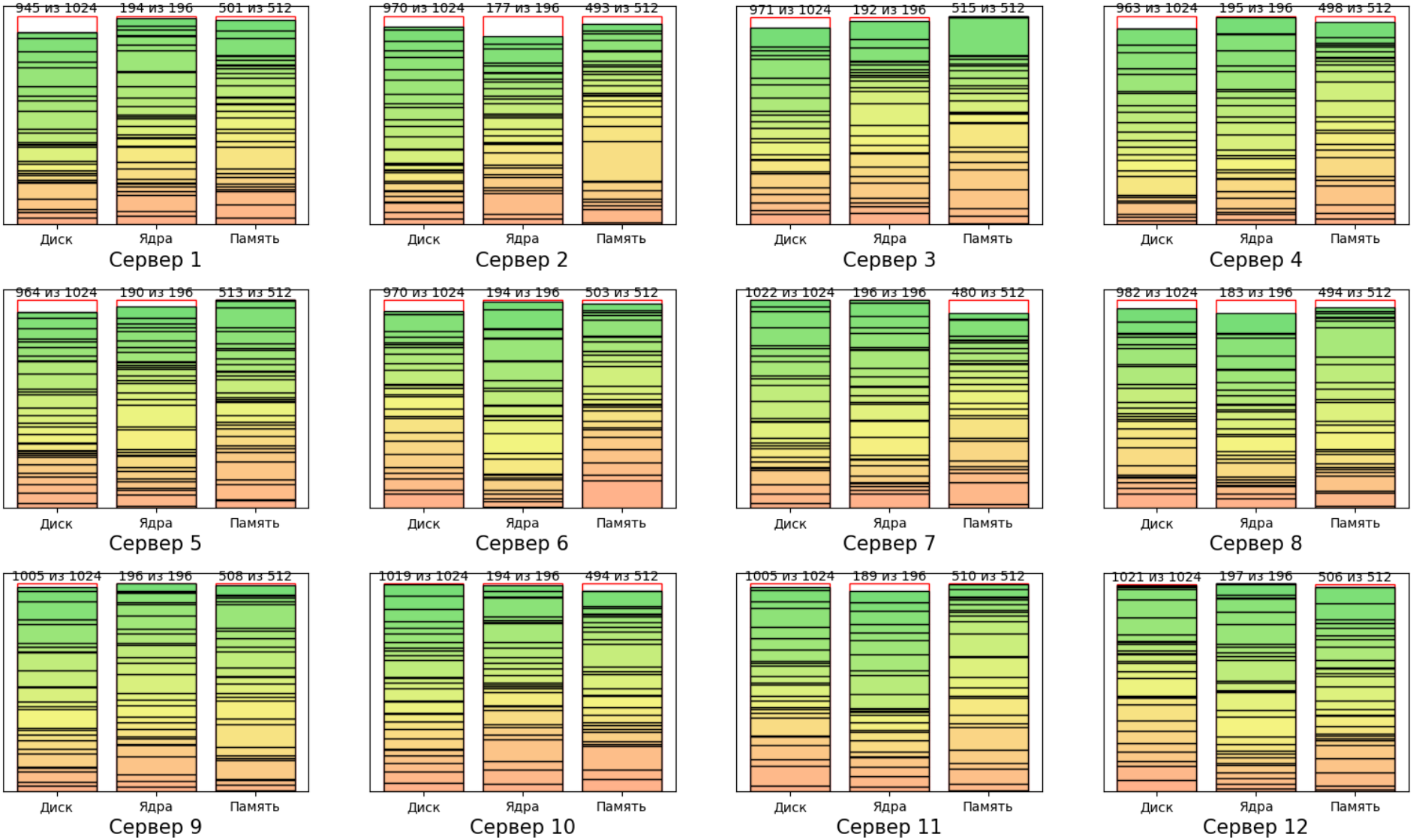
Хорошо видно, что даже для набора из 300 задач ждать приемлемого ответа приходится уже 5 минут.

Хорошо видно, что даже для набора из 300 задач ждать приемлемого ответа приходится уже 5 минут.

А теперь рассмотрим более трудную задачу: 2000 задач и всё те же 12 серверов, только незначительно увеличим ограничение на вместимость по ядрам.

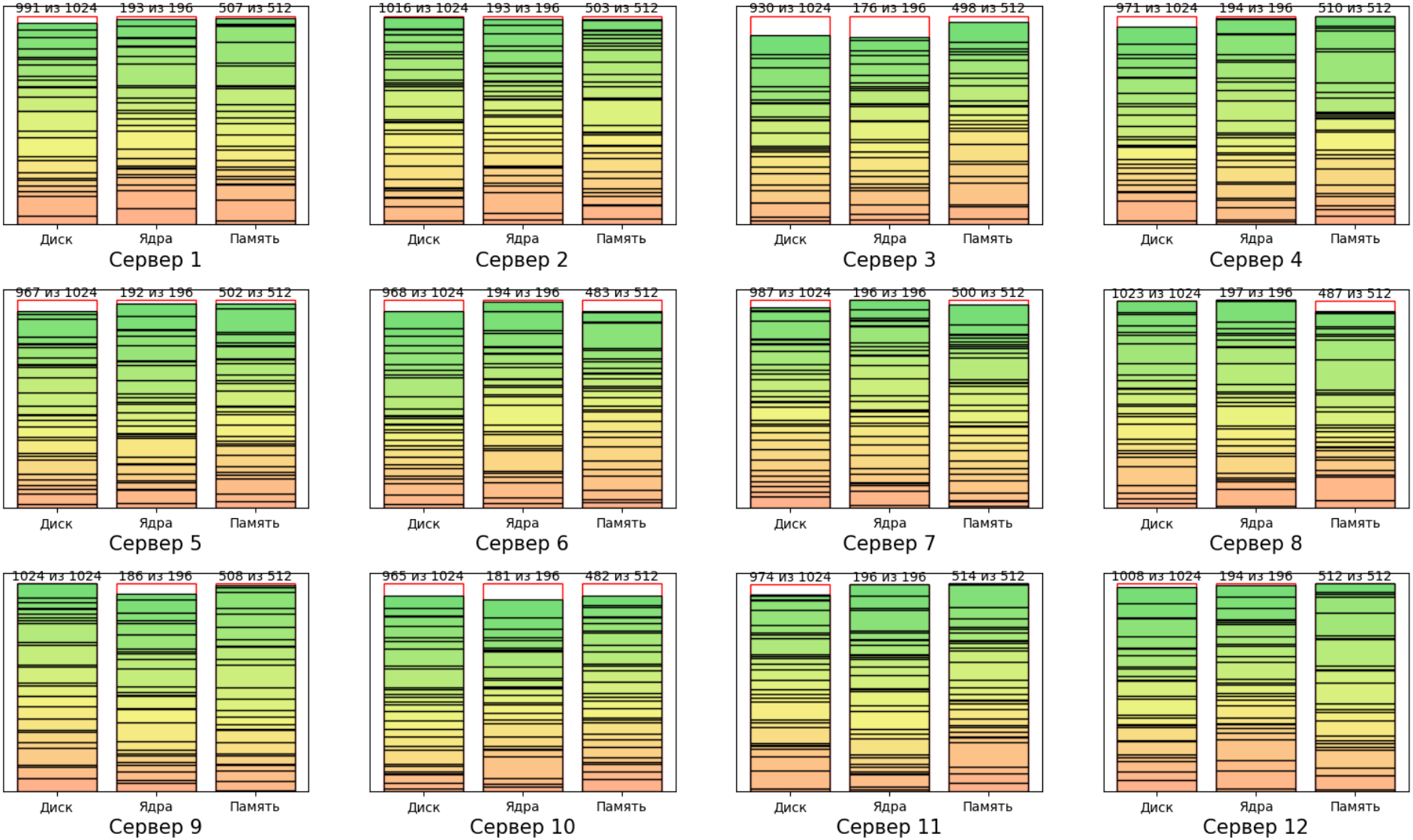
Тест №2 (1). num_reads: 10

Распределение ресурсов на сервере. Количество задач: 2001. Значение num_reads: 10. Время работы: 451.767 s.



Тест №2 (2). num_reads: 100

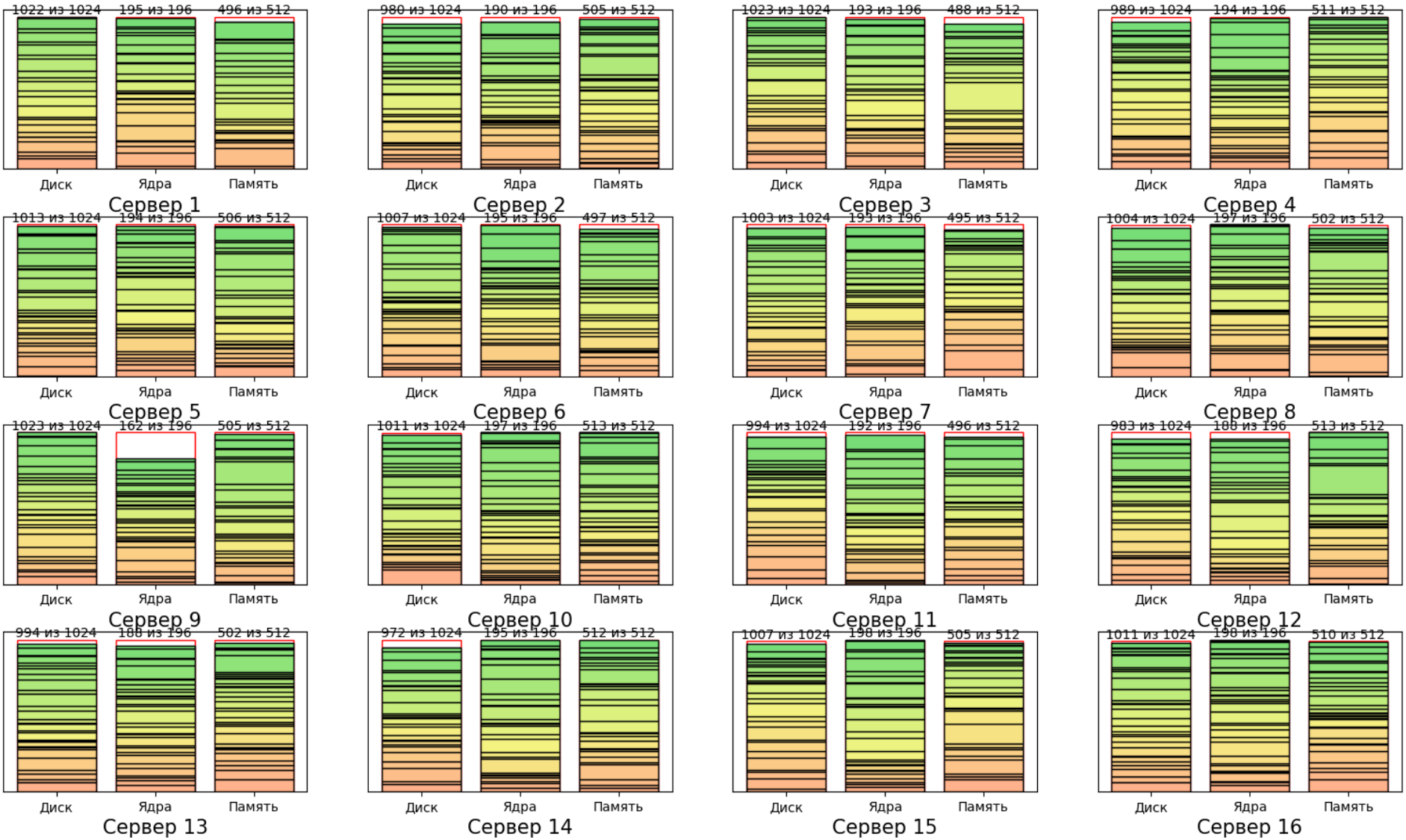
Распределение ресурсов на сервере. Количество задач: 2001. Значение num_reads: 100. Время работы: 1922.635 s.



А что если 16 серверов?

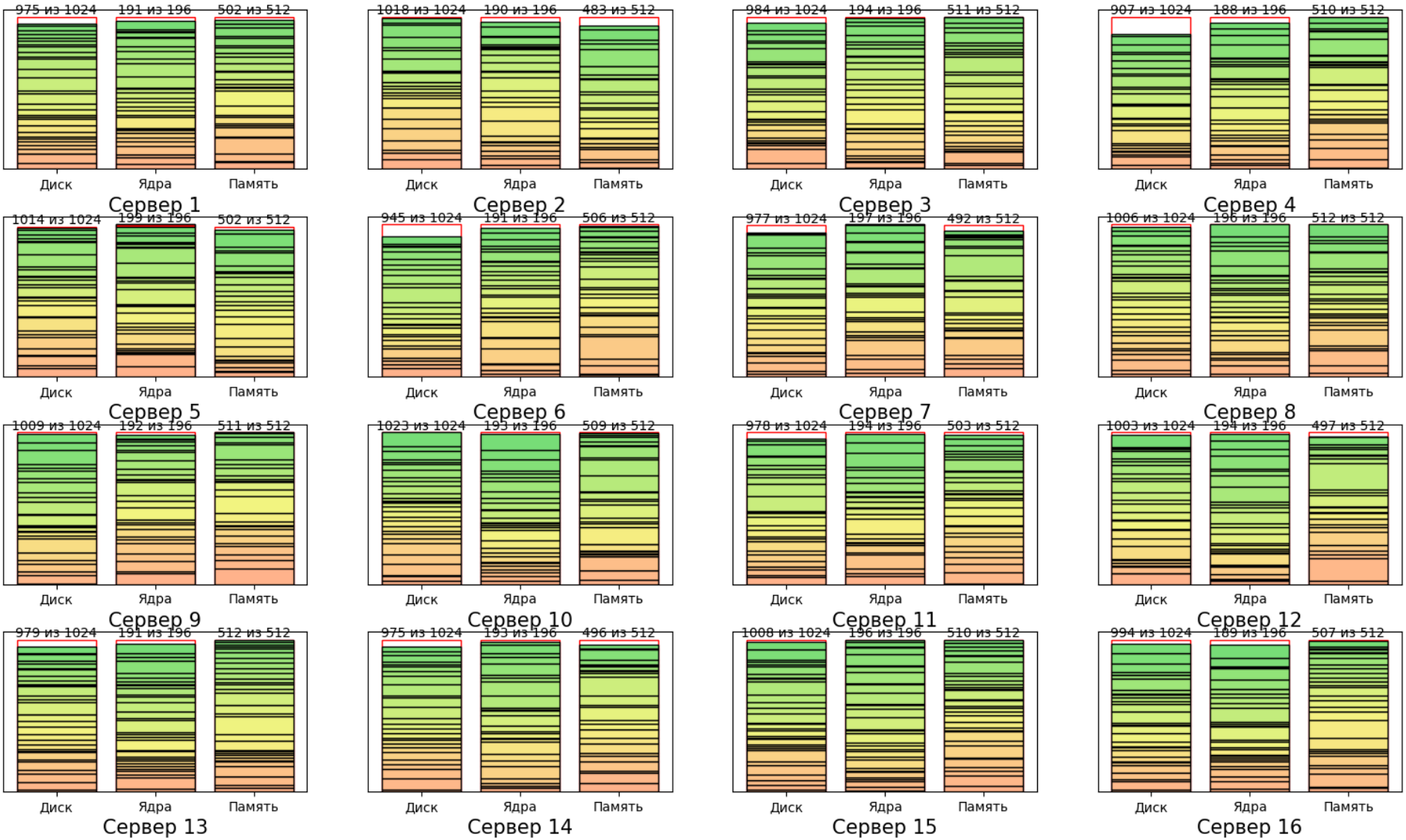
Тест №3 (1). num_reads: 10

Распределение ресурсов на сервере. Количество задач: 2001. Значение num_reads: 100. Время работы: 2587.031 s.



Тест №3 (2). num_reads: 100

Распределение ресурсов на сервере. Количество задач: 2001. Значение num_reads: 100. Время работы: 3075.365 s.



Общий вывод

Замечание

Сложность решения задачи с многими серверами полным перебором составляет $\mathcal{O}(M^N)$, где M – количество серверов, N – количество задач

Общий вывод

Замечание

Сложность решения задачи с многими серверами полным перебором составляет $\mathcal{O}(M^N)$, где M – количество серверов, N – количество задач

Можно видеть, что с увеличение количества задач и серверов сложность задачи растёт очень быстро, что подчёркивает остающуюся NP сложность.

Общий вывод

Замечание

Сложность решения задачи с многими серверами полным перебором составляет $\mathcal{O}(M^N)$, где M – количество серверов, N – количество задач

Можно видеть, что с увеличение количества задач и серверов сложность задачи растёт очень быстро, что подчёркивает остающуюся NP сложность.

Ключевым моментом являются высокие требования данного решения к оперативной памяти на компьютере.

Общий вывод

Замечание

Сложность решения задачи с многими серверами полным перебором составляет $\mathcal{O}(M^N)$, где M – количество серверов, N – количество задач

Можно видеть, что с увеличение количества задач и серверов сложность задачи растёт очень быстро, что подчёркивает остающуюся NP сложность.

Ключевым моментом являются высокие требования данного решения к оперативной памяти на компьютере.

Реализованное решение чувствительно относительно рассматриваемой задачи и сервера, поэтому все перечисленные для них [задач] характеристики могут быть различны на каждом из серверов.

Формулировка проблемы

Формулировка проблемы

Дано $6000 \leqslant N \leqslant 10000$ задач, $500 \leqslant M \leqslant 600$ серверов. Задача j имеет следующие характеристики: α_j – потребление памяти диска, β_j – потребление оперативной памяти, γ_j – количество ядер, которое требуется на выполнение задачи, τ_j – некоторые обобщённые требования, а общее время выполнения задачи есть величина $t = \tau_j / p_i$ на i сервере. Каждый i сервер имеет ограничения по памяти диска – A_i, по оперативной памяти – B_i, по количеству ядер – C_i, а также имеет некоторую производительность p_i и стоимость минуты работы на этом сервере – $cost_i$.

Требования

Требования

Необходимо реализовать алгоритм, который

- Максимально утилизирует ресурсы
- Время простоя ресурсов должно быть минимально
- Время выполнения всех задач должно быть минимально
- Стоимость выполнения задач должно быть минимальным, но при этом есть ограничение на общее время работы - T

Дополнительные условия:

- Время старта / восстановления задачи - t_{start}
- Время завершения / засыпания задачи - t_{stop}
- Время перемещения задачи на другой сервер - t_{move}

Замечания

С одной стороны, минимальная оценка на количество кубитов (т. е. переменных, которые будут использоваться в матрице) – 3 015 000, если мы рассматриваем лишь заполнение серверов, не говоря о дополнительных условиях на время.

Замечания

С одной стороны, минимальная оценка на количество кубитов (т. е. переменных, которые будут использоваться в матрице) – 3 015 000, если мы рассматриваем лишь заполнение серверов, не говоря о дополнительных условиях на время.

С другой стороны, оперативной памяти, чтобы сохранить матрицу (если считать, что каждая ячейка занимает 1 байт), потребуется 8381,9 ГБ.

Общее заключение:

Дополнительный раздел

[Доп.] Решение на квантовом компьютере

Квантовые компьютеры DWave, к примеру, умеют решать лишь одну задачу – нахождение минимума энергии гамильтониана, поэтому для получения ответа на какую-то проблему необходимо интерпретировать условие в матрицу QUBO.

[Доп.] Решение на квантовом компьютере

Квантовые компьютеры DWave, к примеру, умеют решать лишь одну задачу – нахождение минимума энергии гамильтониана, поэтому для получения ответа на какую-то проблему необходимо интерпретировать условие в матрицу QUBO.

QUBO (Quadratic unconstrained binary optimization) формулировка

$$f_Q(x) = x^T Q x = \sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j, \text{ где } x_i \in \{0, 1\}$$

[Доп.] Приходим к QUBO...

Необходимо понять, как свести данную постановку [рюкзак] к матрице QUBO.

[Доп.] Приходим к QUBO...

Необходимо понять, как свести данную постановку [рюкзак] к матрице QUBO.

Из определения видно, что нам нужно составить матрицу Q, следовательно, использовать неравенства мы не можем.

[Доп.] Приходим к QUBO...

Во-первых, начнём с самой QUBO-формулировки:

$$\min_x x^T Q x = \min_x \left(\sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j\right) = \dots$$

[Доп.] Приходим к QUBO...

Во-первых, начнём с самой QUBO-формулировки:

$$\min_x x^T Q x = \min_x \left(\sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j\right) = \dots$$

Заметим, что в силу определений $x_i x_i \equiv x_i$:

$$\cdots = \min_x \left(\sum_{i=1}^{N} Q_{ii} x_i + \sum_{i=1}^{N} \sum_{j>i}^{N} Q_{ij} x_i x_j\right)$$

[Доп.] Приходим к QUBO...

Во-первых, начнём с самой QUBO-формулировки:

$$\min_x x^T Q x = \min_x \left(\sum_{i=1}^{N} \sum_{j \geqslant i}^{N} Q_{ij} x_i x_j\right) = \dots$$

Заметим, что в силу определений $x_i x_i \equiv x_i$:

$$\dots = \min_x \left(\sum_{i=1}^{N} Q_{ii} x_i + \sum_{i=1}^{N} \sum_{j>i}^{N} Q_{ij} x_i x_j\right)$$

Задача поиска минимума гамильтониана эквивалентна нахождению минимума $f_Q(x) = x^T Q x$, где Q – матрица QUBO.

[Доп.] Приходим к QUBO...

Обозначения:

- x_i – переменная регистра. $x_i = 1 \Leftrightarrow i$ предмет находится в рюкзаке
- w_i – вес i-го предмета
- c_i – стоимость i-го предмета
- N – общее количество предметов
- W – максимальный вес, который можно положить в рюкзак

[Доп.] Приходим к QUBO...

Заметим, что мы умеем находить минимальную энергию гамильтониана, а по условиям проблемы необходима максимальная стоимость.

[Доп.] Приходим к QUBO...

Заметим, что мы умеем находить минимальную энергию гамильтониана, а по условиям проблемы необходима максимальная стоимость.

Но если умеем находить минимум, то легко найдём и максимум, рассмотрев $f_{-Q}(x) = -f_Q(x)$.

[Доп.] Приходим к QUBO...

Как было оговорено ранее, неравенств у нас нет. Для их «имитации» введём понятие пенальти.

[Доп.] Приходим к QUBO...

Как было оговорено ранее, неравенств у нас нет. Для их «имитации» введём понятие пенальти.

То есть если мы набрали больше, чем нужно, то получаем какой-то большой штраф, показывая таким образом, что такой выбор был не оптимален.

[Доп.] Приходим к QUBO...

Введём вспомогательную функцию, отслеживающую переполнение.

[Доп.] Приходим к QUBO...

Введём вспомогательную функцию, отслеживающую переполнение.

Рассмотрим её как $\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W$, где y_k – **дополнительные** переменные регистра, отличные от x_i.

[Доп.] Приходим к QUBO...

Иными словами, слагаемое $\sum_{k=1}^{W} k y_k$ будет равняться значению, которое необходимо набрать, чтобы получить вес W.

[Доп.] Приходим к QUBO...

Иными словами, слагаемое $\sum_{k=1}^{W} k y_k$ будет равняться значению, которое необходимо набрать, чтобы получить вес W.

Тогда мы сможем сказать следующее: $\rho(x) = 0 \Leftrightarrow \sum_{i=1}^{N} w_i x_i \leqslant W$, а в остальных случаях $\rho(x) \neq 0$.

Таким образом, при переполнении рюкзака получаем слишком большое отклонение – пенальти (штраф).

[Доп.] Приходим к QUBO...

Итого, построили такую функцию для пенальти:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W.$$

[Доп.] Приходим к QUBO...

Итого, построили такую функцию для пенальти:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W.$$

Казалось бы, уже хорошо, но можно лучше: сократим количество слагаемых в вспомогательной функции $\rho(x)$ до $N + \lfloor \log_2 W \rfloor + 1$.

[Доп.] Приходим к QUBO...

Итого, построили такую функцию для пенальти:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W.$$

Казалось бы, уже хорошо, но можно лучше: сократим количество слагаемых в вспомогательной функции $\rho(x)$ до $N + \lfloor \log_2 W \rfloor + 1$.

[Доп.] Приходим к QUBO...

Итого, построили такую функцию для пенальти:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=1}^{W} k y_k - W.$$

Казалось бы, уже хорошо, но можно лучше: сократим количество слагаемых в вспомогательной функции $\rho(x)$ до $N + \lfloor \log_2 W \rfloor + 1$.

Тогда имеем следующую функцию, где уже представлены **дополнительные** переменные регистра y'_k, отличные от y_k и x_i:

$$\rho(x) = \sum_{i=1}^{N} w_i x_i + \sum_{k=0}^{\lfloor \log_2 W \rfloor} 2^k y'_k - W$$

[Доп.] Приходим к QUBO...

Подставим $\rho(x)$ в $f_Q(x)$:

$$f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{N} w_i x_i + \sum_{k=0}^{\lfloor \log_2 W \rfloor} 2^k y'_k - W)^2$$

Для удобства записи будем считать, что $k = i - N - 1$: $y'_k = x_i$ и $2^k = w_i \ \forall i > N$.

Эта договорённость сделает форму записи более компактной:

$$f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor + 1$$

[Доп.] Приходим к QUBO...

В итоге имеем следующую функцию, максимум которой будем искать:

$$f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor + 1$$

[Доп.] Приходим к QUBO...

В итоге имеем следующую функцию, максимум которой будем искать:

$$f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor + 1$$

Продолжаем упрощать:

$$(\sum_{i=1}^{M} w_i x_i - W)^2 = \sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j x_i x_j - 2W\sum_{i=1}^{M} w_i x_i + W^2 =$$

[Доп.] Приходим к QUBO...

В итоге имеем следующую функцию, максимум которой будем искать:

$$f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor + 1$$

Продолжаем упрощать:

$$(\sum_{i=1}^{M} w_i x_i - W)^2 = \sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j x_i x_j - 2W\sum_{i=1}^{M} w_i x_i + W^2 =$$

$$2\sum_{i=1}^{M}\sum_{j>i}^{M} w_i w_j x_i x_j - \sum_{i=1}^{M}(2W - w_i) w_i x_i + W^2$$

[Доп.] Приходим к QUBO...

В итоге имеем следующую функцию, максимум которой будем искать:

$$f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor + 1$$

Продолжаем упрощать:

$$(\sum_{i=1}^{M} w_i x_i - W)^2 = \sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j x_i x_j - 2W\sum_{i=1}^{M} w_i x_i + W^2 =$$
$$2\sum_{i=1}^{M}\sum_{j>i}^{M} w_i w_j x_i x_j - \sum_{i=1}^{M}(2W - w_i) w_i x_i + W^2$$

Тогда элементы матрицы Q:

[Доп.] Приходим к QUBO...

В итоге имеем следующую функцию, максимум которой будем искать:

$$f_Q(x) = -\sum_{i=1}^{N} c_i x_i + \lambda(\sum_{i=1}^{M} w_i x_i - W)^2, \text{ где } M = N + \lfloor \log_2 W \rfloor + 1$$

Продолжаем упрощать:

$$(\sum_{i=1}^{M} w_i x_i - W)^2 = \sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j x_i x_j - 2W\sum_{i=1}^{M} w_i x_i + W^2 =$$
$$2\sum_{i=1}^{M}\sum_{j>i}^{M} w_i w_j x_i x_j - \sum_{i=1}^{M}(2W - w_i) w_i x_i + W^2$$

Тогда элементы матрицы Q:

$$\begin{cases} Q_{ii} &= -c_i - \lambda w_i(2W - w_i) \\ Q_{ij} &= 2\lambda w_i w_j \end{cases}$$

[Доп.] MCSKS. Формулировка проблемы.

Формулировка проблемы в терминах задач и серверов

Дано N задач, i задача имеет следующие параметры: потребление памяти диска $\alpha_i > 0$, потребление оперативной памяти $\beta_i > 0$ и требует ядер $\gamma_i > 0$. Необходимо выбрать такое подмножество задач, чтобы их суммарное потребление памяти на диске не превосходило A, количество ядер – B, а потребление оперативной памяти – C, но при этом распределение ресурсов было максимально.

[Доп.] MCSKS. Приходим к QUBO...

Введём обозначения:

[Доп.] MCSKS. Приходим к QUBO...

Введём обозначения:

- x_i – переменная регистра. $x_i = 1 \Leftrightarrow i$ задача на сервере.

[Доп.] MCSKS. Приходим к QUBO...

Введём обозначения:

- x_i – переменная регистра. $x_i = 1 \Leftrightarrow i$ задача на сервере.
- α_i – память диска, которую потребляет i задача

[Доп.] MCSKS. Приходим к QUBO...

Введём обозначения:

- x_i – переменная регистра. $x_i = 1 \Leftrightarrow i$ задача на сервере.
- α_i – память диска, которую потребляет i задача
- β_i – оперативная память, которую потребляет i задача

[Доп.] MCSKS. Приходим к QUBO...

Введём обозначения:

- x_i – переменная регистра. $x_i = 1 \Leftrightarrow i$ задача на сервере.
- α_i – память диска, которую потребляет i задача
- β_i – оперативная память, которую потребляет i задача
- γ_i – количество ядер, потребляемых i задачей

[Доп.] MCSKS. Приходим к QUBO...

Введём обозначения:

- x_i – переменная регистра. $x_i = 1 \Leftrightarrow i$ задача на сервере.
- α_i – память диска, которую потребляет i задача
- β_i – оперативная память, которую потребляет i задача
- γ_i – количество ядер, потребляемых i задачей
- a_k – дополнительные переменные регистра для контроля потребления памяти диска задачами

[Доп.] MCSKS. Приходим к QUBO...

Введём обозначения:

- x_i – переменная регистра. $x_i = 1 \Leftrightarrow i$ задача на сервере.
- α_i – память диска, которую потребляет i задача
- β_i – оперативная память, которую потребляет i задача
- γ_i – количество ядер, потребляемых i задачей
- a_k – дополнительные переменные регистра для контроля потребления памяти диска задачами
- b_k – дополнительные переменные регистра для контроля потребления количества ядер задачами

[Доп.] MCSKS. Приходим к QUBO...

Введём обозначения:

- x_i – переменная регистра. $x_i = 1 \Leftrightarrow i$ задача на сервере.
- α_i – память диска, которую потребляет i задача
- β_i – оперативная память, которую потребляет i задача
- γ_i – количество ядер, потребляемых i задачей
- a_k – дополнительные переменные регистра для контроля потребления памяти диска задачами
- b_k – дополнительные переменные регистра для контроля потребления количества ядер задачами
- c_k – дополнительные переменные регистра для контроля потребляемой оперативной памяти

[Доп.] MCSKS. Приходим к QUBO...

Тогда функция в ведённых переменных, минимум которой будем искать, примет вид:

[Доп.] MCSKS. Приходим к QUBO...

Тогда функция в ведённых переменных, минимум которой будем искать, примет вид:

$$f_Q(x) = \left[-\sum_{i=1}^{N} \alpha_i x_i - \sum_{i=1}^{N} \beta_i x_i - \sum_{i=1}^{N} \gamma_i x_i \right] +$$
$$+ \lambda_1 \left(\sum_{i=1}^{N} \alpha_i x_i + \sum_{k=0}^{\lfloor \log_2 A \rfloor + 1} 2^k a_k - A \right)^2 +$$
$$+ \lambda_2 \left(\sum_{i=1}^{N} \beta_i x_i + \sum_{k=0}^{\lfloor \log_2 B \rfloor + 1} 2^k b_k - B \right)^2 +$$
$$+ \lambda_3 \left(\sum_{i=1}^{N} \gamma_i x_i + \sum_{k=0}^{\lfloor \log_2 C \rfloor + 1} 2^k c_k - C \right)^2 .$$

[Доп.] MCMKS. Формулировка проблемы

Формулировка в терминах задач и серверов

Дано N задач, M серверов, i задача имеет следующие параметры: потребление памяти диска $\alpha_j > 0$, потребление оперативной памяти $\beta_j > 0$ и требует ядер $\gamma_j > 0$. Необходимо найти такое подмножество задач, чтобы распределение ресурсов по M серверам было максимально, но количество ядер потребляемых совокупностью задач на i сервере не превосходило заданной величины A_i (вместимость в ядрах i-го сервера), по потребляемой памяти диска – B_i (вместимость i-го сервера по памяти диска), а также по оперативной памяти – C_i. Более того, каждая задача может быть запущена не более чем K раз (если не оговорено иного, $K = 1$).

[Доп.] MCMKS. Приходим к QUBO...

Введём обозначения:

[Доп.] MCMKS. Приходим к QUBO...

Введём обозначения:

- x_{ij} – переменная регистра. $x_{ij} = 1 \Leftrightarrow j$ задача на i сервере.

[Доп.] MCMKS. Приходим к QUBO...

Введём обозначения:

- x_{ij} – переменная регистра. $x_{ij} = 1 \Leftrightarrow j$ задача на i сервере.
- α_{ij} – память диска, которую потребляет j задача на i сервере

[Доп.] MCMKS. Приходим к QUBO...

Введём обозначения:

- x_{ij} – переменная регистра. $x_{ij} = 1 \Leftrightarrow j$ задача на i сервере.
- α_{ij} – память диска, которую потребляет j задача на i сервере
- β_{ij} – оперативная память, которую потребляет j задача на i сервере

[Доп.] MCMKS. Приходим к QUBO...

Введём обозначения:

- x_{ij} – переменная регистра. $x_{ij} = 1 \Leftrightarrow j$ задача на i сервере.
- α_{ij} – память диска, которую потребляет j задача на i сервере
- β_{ij} – оперативная память, которую потребляет j задача на i сервере
- γ_{ij} – количество ядер, потребляемых j задачей на i сервере

[Доп.] MCMKS. Приходим к QUBO...

Введём обозначения:

- x_{ij} – переменная регистра. $x_{ij} = 1 \Leftrightarrow j$ задача на i сервере.
- α_{ij} – память диска, которую потребляет j задача на i сервере
- β_{ij} – оперативная память, которую потребляет j задача на i сервере
- γ_{ij} – количество ядер, потребляемых j задачей на i сервере
- a_{pk} – дополнительные переменные регистра для контроля потребления памяти диска задачами на p-м сервере

[Доп.] MCMKS. Приходим к QUBO...

Введём обозначения:

- x_{ij} – переменная регистра. $x_{ij} = 1 \Leftrightarrow j$ задача на i сервере.
- α_{ij} – память диска, которую потребляет j задача на i сервере
- β_{ij} – оперативная память, которую потребляет j задача на i сервере
- γ_{ij} – количество ядер, потребляемых j задачей на i сервере
- a_{pk} – дополнительные переменные регистра для контроля потребления памяти диска задачами на p-м сервере
- b_{pk} – дополнительные переменные регистра для контроля потребления количества ядер задачами на p-м сервере

[Доп.] MCMKS. Приходим к QUBO...

Введём обозначения:

- x_{ij} – переменная регистра. $x_{ij} = 1 \Leftrightarrow j$ задача на i сервере.
- α_{ij} – память диска, которую потребляет j задача на i сервере
- β_{ij} – оперативная память, которую потребляет j задача на i сервере
- γ_{ij} – количество ядер, потребляемых j задачей на i сервере
- a_{pk} – дополнительные переменные регистра для контроля потребления памяти диска задачами на p-м сервере
- b_{pk} – дополнительные переменные регистра для контроля потребления количества ядер задачами на p-м сервере
- c_{pk} – дополнительные переменные регистра для контроля потребляемой оперативной памяти на p-м сервере

[Доп.] MCMKS. Приходим к QUBO...

Опуская подробности, получаем:

[Доп.] MCMKS. Приходим к QUBO...

Опуская подробности, получаем:

$$
\begin{aligned}
f_Q(x) = & -\lambda_1 \cdot \sum_{i=1}^{N}\sum_{j=1}^{M}(\alpha_{ij}+\beta_{ij}+\gamma_{ij})\cdot x_{ij} + \\
& + \lambda_2 \cdot \sum_{i=1}^{M}\left[\left(\sum_{j=1}^{N}\alpha_{ij}\cdot x_{ij}\right) + \left(\sum_{b=0}^{\lfloor \log_2 A_i \rfloor} 2^b \cdot \widetilde{a}_{ib}\right) - A_i\right]^2 + \\
& + \lambda_3 \cdot \sum_{i=1}^{M}\left[\left(\sum_{j=1}^{N}\beta_{ij}\cdot x_{ij}\right) + \left(\sum_{b=0}^{\lfloor \log_2 B_i \rfloor} 2^b \cdot \widetilde{b}_{ib}\right) - B_i\right]^2 + \\
& + \lambda_4 \cdot \sum_{i=1}^{M}\left[\left(\sum_{j=1}^{N}\gamma_{ij}\cdot x_{ij}\right) + \left(\sum_{b=0}^{\lfloor \log_2 C_i \rfloor} 2^b \cdot \widetilde{c}_{ib}\right) - C_i\right]^2 + \\
& + \lambda_5 \cdot \sum_{j=1}^{N}\prod_{k=0}^{K}\sum_{i=1}^{M}(x_{ij}-k)
\end{aligned}
$$

Конец