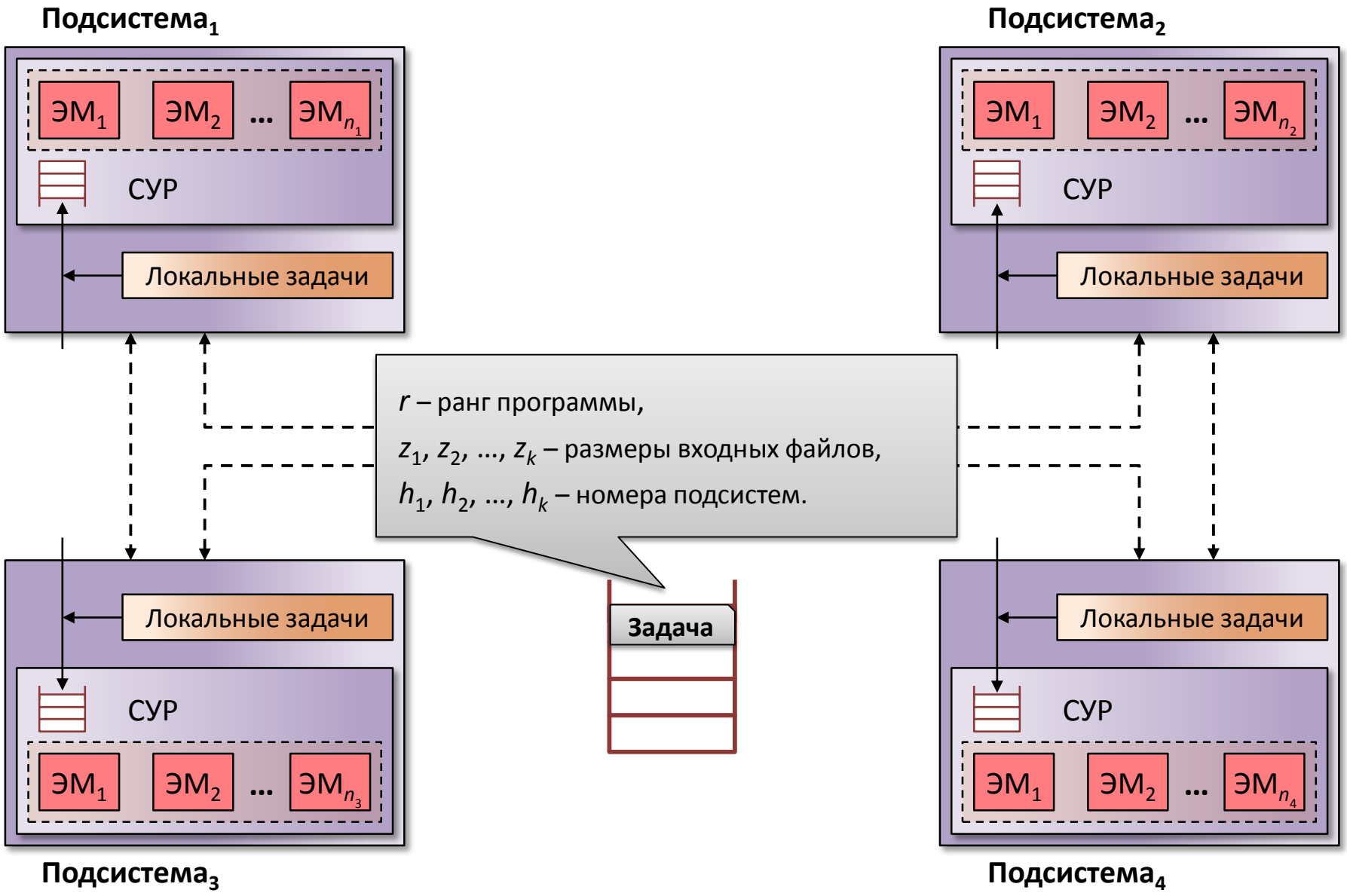
Лекция 10. Децентрализованная диспетчеризация

Пазников Алексей Александрович

к.т.н., ст. преп. Кафедры вычислительных систем
Сибирский государственный университет
телекоммуникаций и информатики

http://cpct.sibsutis.ru/~apaznikov

Диспетчеризация в пространственно-распределённых ВС



Диспетчеризация в пространственно-распределённых ВС

Пусть имеется мультикластерная ВС, состоящая из H подсистем,

c_j – количество свободных ЭМ в подсистеме j;

q_j – количество задач в состоянии ожидания;

$w_j = q_j / n_j$ – количество задач в очереди, приходящееся на одну ЭМ;

$t_j = \sum_{l=1}^{k} t(h_l, j, z_l)$ – оценка времени доставки файлов задачи до подсистемы j

Требуется отыскать подсистему j^*, в которой достигается минимум функции $F(j)$

$$j^* = \arg\min_{j \in L(i) \cup \{i\}} \{F(j)\}$$

$$F(j) = \begin{cases} \dfrac{t_j}{t_{\max}} + \dfrac{c_j^{-1}}{c_{\max}^{-1}} + \dfrac{w_j}{w_{\max}}, & \text{если } c_j < r \text{ или } q_j > 0, \\ \dfrac{t_j}{t_{\max}}, & \text{иначе.} \end{cases}$$

где $t_{\max} = \max_{j \in S(i)} \{t_j\}$, $c_{\max} = \max_{j \in S(i)} \{c_j\}$, $w_{\max} = \max_{j \in S(i)} \{w_j\}$

Диспетчеризация в пространственно-распределённых ВС

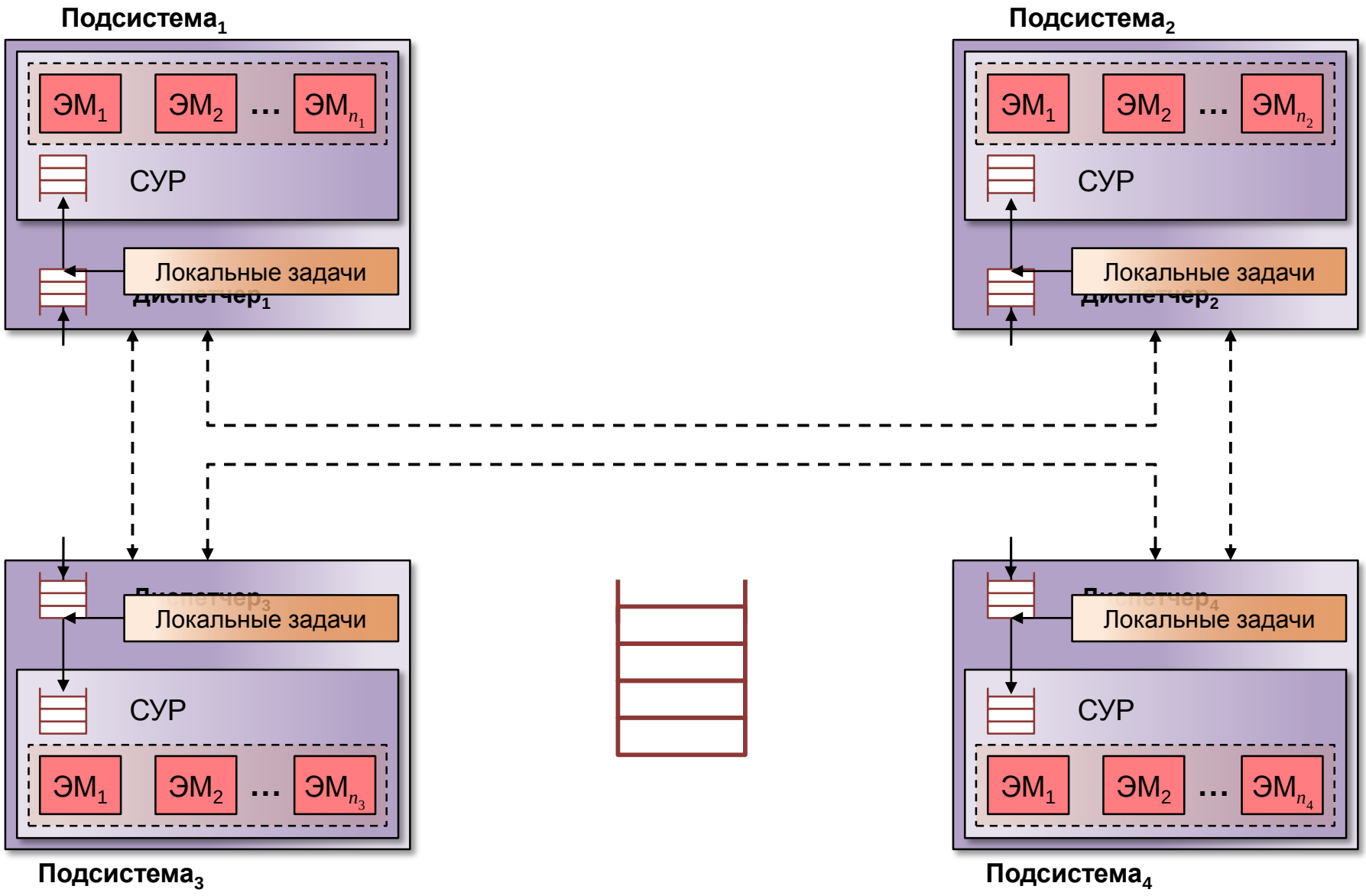
Централизованная диспетчеризация

- Buyya, Abramson (*Nimrod/G*), 2000
- Frey, Tannenbaum (*Condor-G*), 2001
- Berman, Wolski (*AppLeS*), 2003
- Mcgough, Young (*ICENI*), 2004
- Huedo, Montero (*GridWay*), 2004
- Cooper, Dasgupta (*GrADS*), 2004
- Andretto, Borgia (*WMS*), 2004
- Deelman, Singh (*Pegasus*), 2005
- Fahringer, Prodan (*ASKALON*), 2005
- David, Gombas (*Triana*), 2006
- Missier, Soiland-Reyes (*Taverna*), 2010

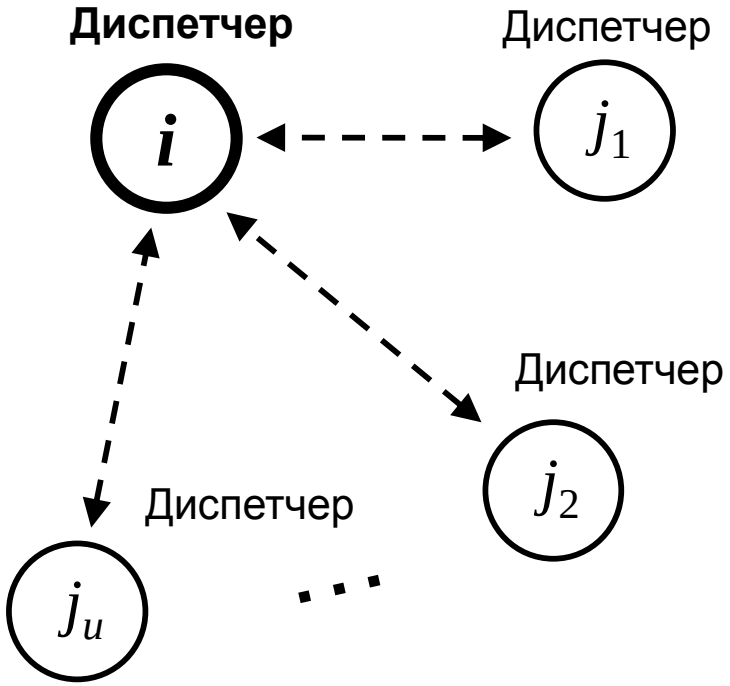
Децентрализованная диспетчеризация

- Корнеев, 1985
- Монахов, 2000
- Wijngaards, Overeinder, 2002
- Gradwell (*AgentSpace*), 2003
- Normale, Lyon (*DIRAC*), 2005
- Huang, Brocco (*SmartGRID*), 2008
- Alexander, Grimme, 2009
- Altameem, Amoon, 2010
- Solar, Rojas, 2012

Диспетчеризация в пространственно-распределённых ВС



Диспетчеризация в пространственно-распределённых ВС



$$L(i) = \{ j_1, j_2, \ldots, j_u \}$$

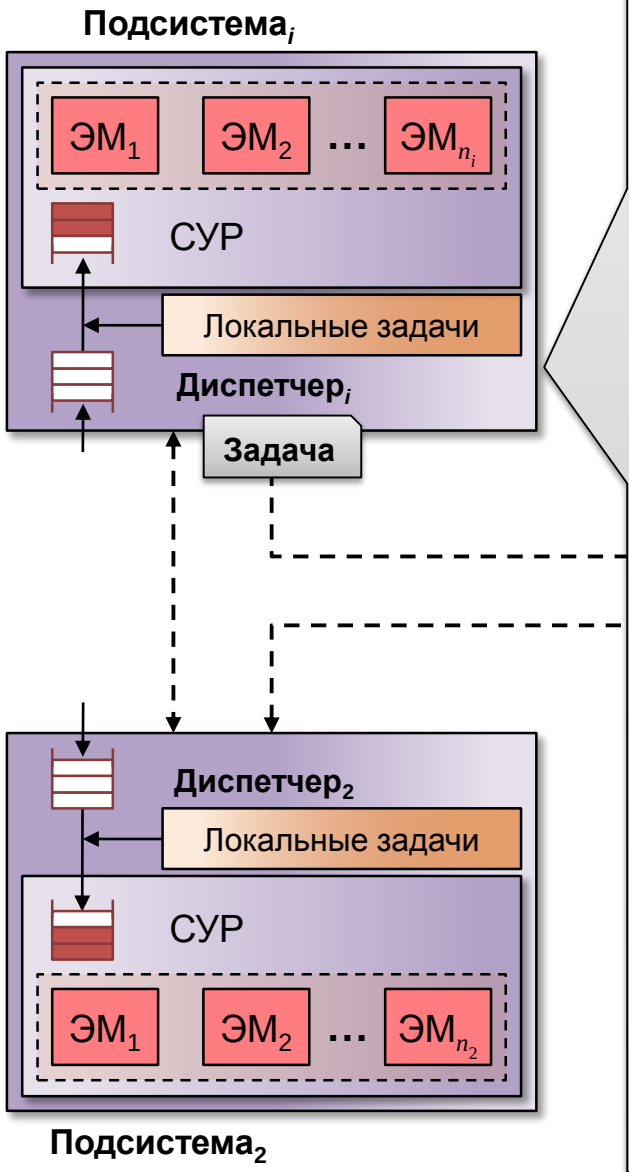
Созданы алгоритмы:

- **Локально-оптимальной диспетчеризации (ДЛО),**
- **На основе репликации задач (ДР),**
- **На основе миграции задач (ДМ),**
- **На основе репликации и миграции задач (ДРМ).**

Каждый алгоритм описывает функционирование диспетчера i при поступлении задачи в его очередь.

Шаг 0: У системы мониторинга запрашиваются значения параметров t_{ij}, c_j, s_j, q_j и n_j, строится множество $S(i) = \{j \mid n_j \geq r, j \in L(i) \cup \{i\}\}$.

Алгоритм локально-оптимальной диспетчеризации



Шаг 1. Выбирается подсистема j^* с минимальным значанием $F(j), j \in S(j)$

$$F(j) = \begin{cases} \dfrac{t_j}{t_{\max}} + \dfrac{c_j^{-1}}{c_{\max}^{-1}} + \dfrac{w_j}{w_{\max}}, & \text{если } c_j < r \text{ или } q_j > 0, \\ \dfrac{t_j}{t_{\max}}, & \text{иначе.} \end{cases}$$

c_j – количество свободных ЭМ в подсистеме j;

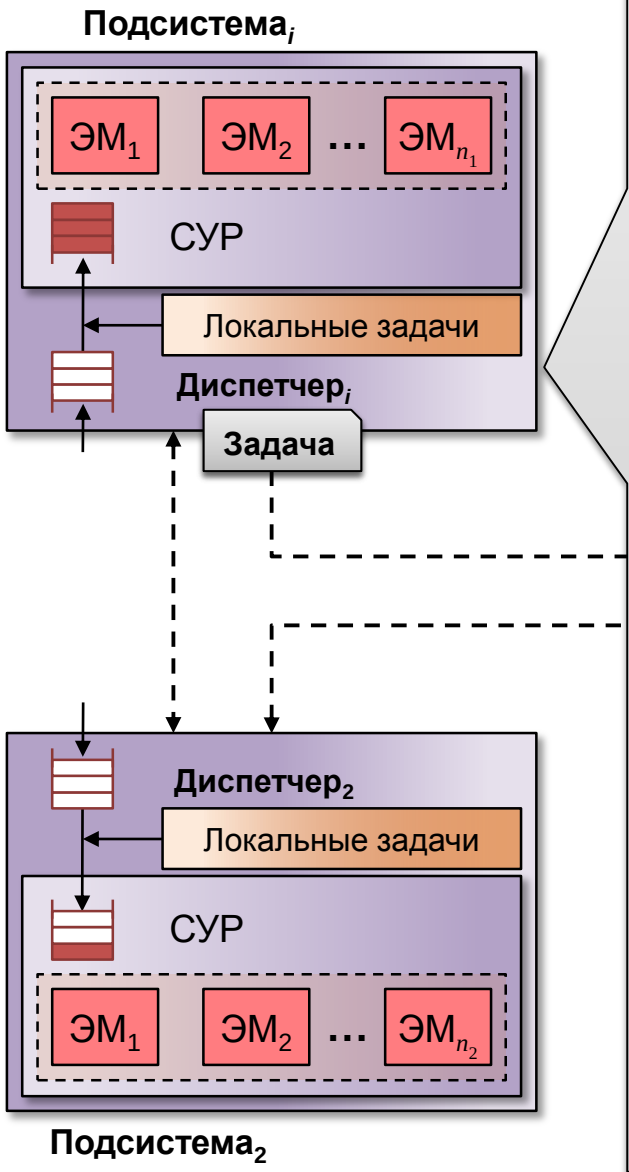
q_j – количество задач в состоянии ожидания;

$w_j = q_j / n_j$ – количество задач в очереди, приходящееся на одну ЭМ;

$t_j = \sum_{l=1}^{k} t(h_l, j, z_l)$ – оценка времени доставки файлов задачи до подсистемы j

Шаг 2. Задача направляется в очередь локальной СУР подсистемы j^*.

Алгоритм на основе репликации задач



Шаг 1. Выбирается m подсистем $j^*_1, j^*_2, \ldots, j^*_m \in S(i)$ в порядке неубывания значений $F(j)$

$$F(j) = \begin{cases} \dfrac{t_j}{t_{\max}} + \dfrac{c_j^{-1}}{c_{\max}^{-1}} + \dfrac{w_j}{w_{\max}}, & \text{если } c_j < r \text{ или } q_j > 0, \\ \dfrac{t_j}{t_{\max}}, & \text{иначе.} \end{cases}$$

c_j – количество свободных ЭМ в подсистеме j;

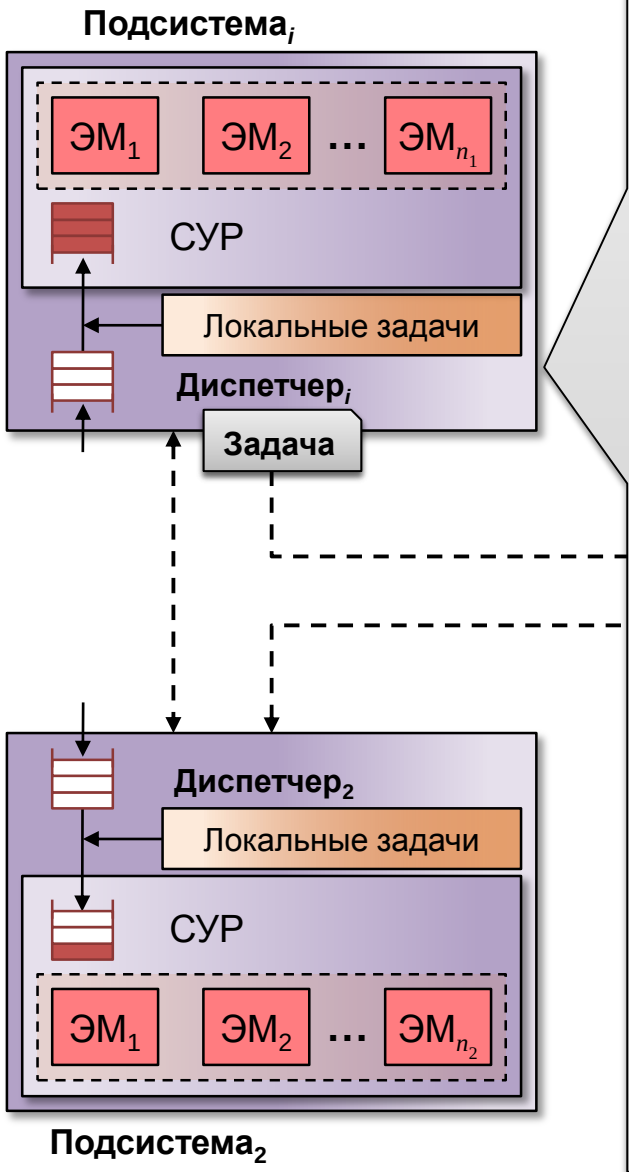
q_j – количество задач в состоянии ожидания;

$w_j = q_j / n_j$ – количество задач в очереди, приходящееся на одну ЭМ;

$t_j = \sum_{l=1}^{k} t(h_l, j, z_l)$ – оценка времени доставки данных до подсистемы j

Шаг 2. Задача направляется в очереди подсистем $j^*_1, j^*_2, \ldots, j^*_m$.

Алгоритм на основе миграции задач



Шаг 1. Выбирается m подсистем $j^*_1, j^*_2, \ldots, j^*_m \in S(i)$ в порядке неубывания значений $F(j)$

$$F(j) = \begin{cases} \dfrac{t_j}{t_{\max}} + \dfrac{c_j^{-1}}{c_{\max}^{-1}} + \dfrac{w_j}{w_{\max}}, & \text{если } c_j < r \text{ или } q_j > 0, \\ \dfrac{t_j}{t_{\max}}, & \text{иначе.} \end{cases}$$

c_j – количество свободных ЭМ в подсистеме j;

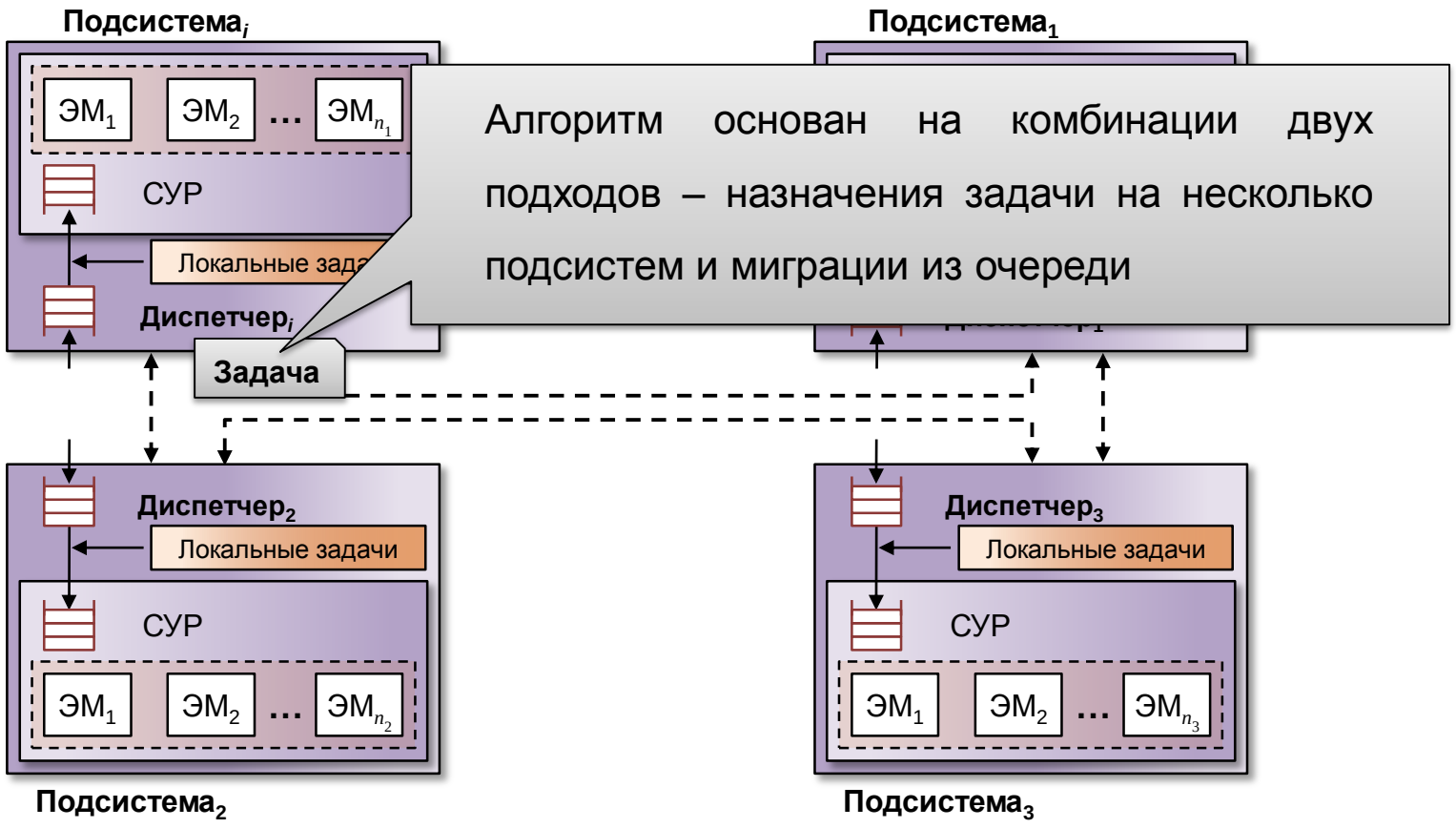
q_j – количество задач в состоянии ожидания;

$w_j = q_j / n_j$ – количество задач в очереди, приходящееся на одну ЭМ;

$t_j = \sum_{l=1}^{k} t(h_l, j, z_l)$ – оценка времени доставки данных до подсистемы j

Шаг 2. Задача направляется в очереди подсистем $j^*_1, j^*_2, \ldots, j^*_m$.

Алгоритм на основе репликации и миграции задач

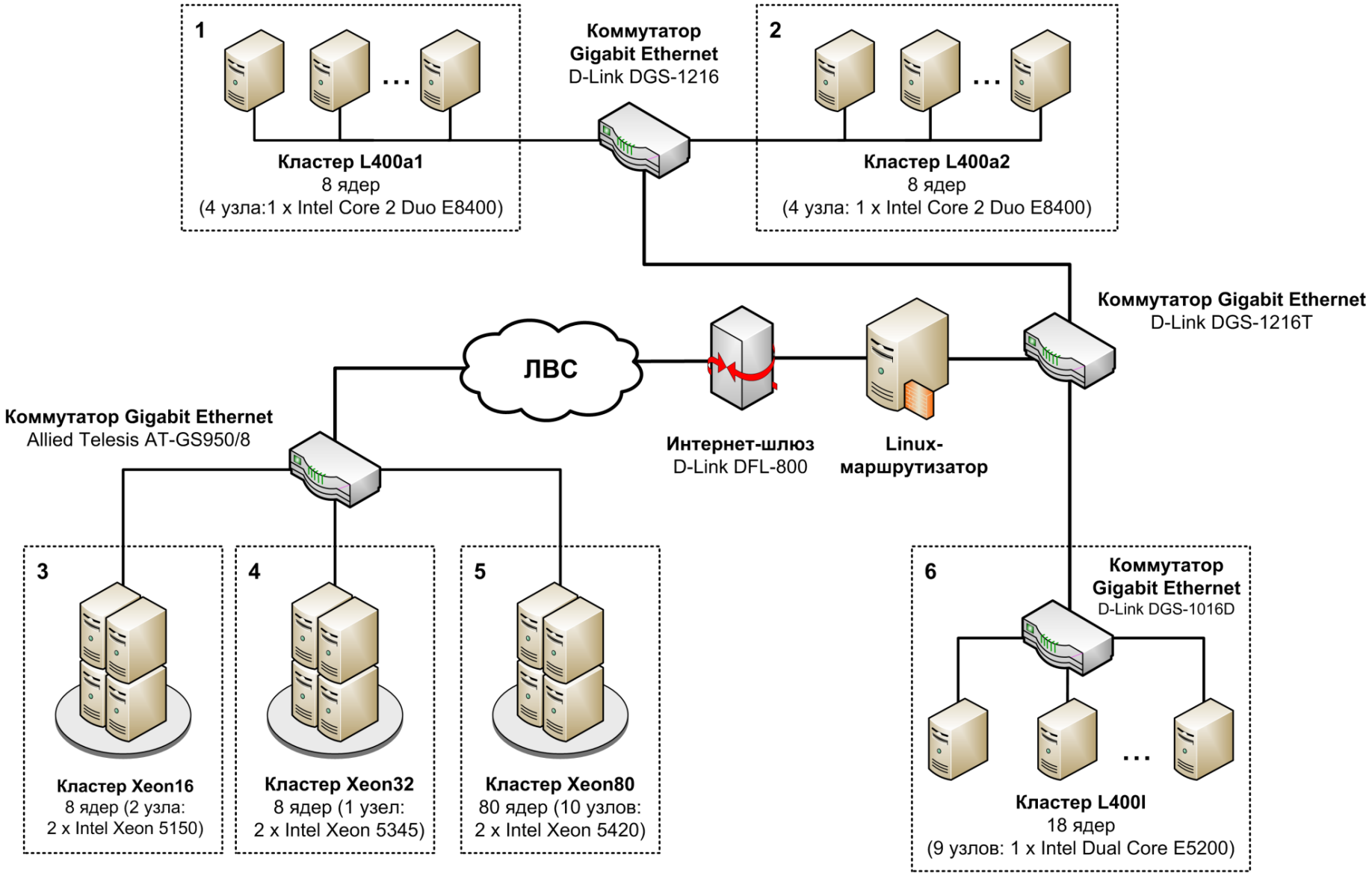


Трудоёмкость алгоритма поиска подсистемы:

$$T = O(|S(i)|\Delta t),$$

где $|\Delta t|$ – время получения информации о производительности подсистемы и времени доставки данных

Тестовая конфигурация мультикластерной ВС



$H = 6$, $N = 130$

Тестовые задачи

Параллельные MPI-программы из пакета SPEC MPI2007:

- **Weather Research and Forecasting (WRF)** – пакет моделирования климатических процессов.
- **The Parallel Ocean Program (POP)** – пакет моделирования процессов в мировом океане.
- **LAMMPS** – пакет решения задач молекулярной динамики.
- **RAxML** – пакет моделирования задач биоинформатики.
- **Tachyon** – пакет расчета графических сцен.

Формирование потоков задач:

- Простейшие потоки задач с интенсивностью λ.
- Задачи выбирались псевдослучайным образом из тестового набора с равномерным распределением.
- Ранг r параллельной программы выбирался из множества $\{1, 2, 4, 8\}$.
- Входные данные задач находились на подсистеме Xeon80.

Показатели эффективности диспетчеризации

- **Пропускная способность *B* системы:**

$$B = M / \tau$$

- **Среднее время *W* пребывания задачи в очереди:**

$$W = \frac{1}{M} \sum_{k=1}^{M} (t'_k - t_k)$$

- **Среднее время *T* обслуживания задачи:**

$$T = \frac{1}{M} \sum_{k=1}^{M} (t''_k - t_k)$$

Обозначения:

- t_k – время поступления задачи $k \in \{1, 2, \ldots, M\}$ на вход диспетчера.
- t'_k – время запуска задачи k на выполнение.
- t''_k – время завершения выполнения задачи k.

Показатели эффективности диспетчеризации

Использование алгоритмов

- Небольшие входные данные $\Rightarrow$ **ДЛО, ДМ, ДР, ДРМ.**
- Большие входные данные $\Rightarrow$ **ДЛО и ДМ.**
- Малая интенсивность потоков

  или небольшие входные файлы $\Rightarrow$ **ДР и ДРМ**

ДЛО – алгоритм локально-оптимальной диспетчеризации,

ДР – алгоритм на основе репликации задач,

ДМ – алгоритм на основе миграции задач (ДМ),

ДРМ – алгоритм на основе репликации и миграции задач (ДРМ).

Показатели эффективности диспетчеризации

Использование алгоритмов

- Небольшие входные данные $\Rightarrow$ **ДЛО, ДМ, ДР, ДРМ.**
- Большие входные данные $\Rightarrow$ **ДЛО и ДМ.**
- Малая интенсивность потоков

  или небольшие входные файлы $\Rightarrow$ **ДР и ДРМ**

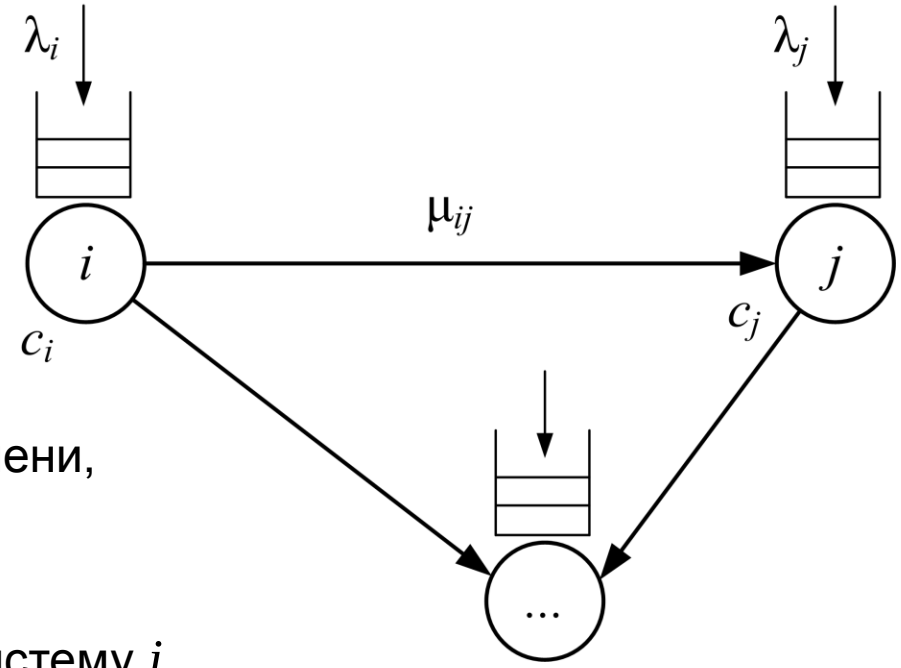
ДЛО – алгоритм локально-оптимальной диспетчеризации,

ДР – алгоритм на основе репликации задач,

ДМ – алгоритм на основе миграции задач (ДМ),

ДРМ – алгоритм на основе репликации и миграции задач (ДРМ).

Формирование субоптимальных локальных окрестностей



H – количество подсистем,

c_i – стоимость использования подсистемы в единицу времени,

λ_i – интенсивность потока поступления задач на подсистему i,

μ_{ij} – интенсивность потока миграции задач с i на j.

$x_{ij} = \{0, 1\}$ – наличие дуги от i к j.

Формирование субоптимальных локальных окрестностей

$\sum_{j=1..H} x_{ij}\mu_{ij}$ – количество задач, мигрировавших с подсистемы i,

$\sum_{j=1..H} x_{ij}\mu_{ij}c_j(t_i + kt_i)$ – стоимость обслуживания мигрировавших задач,

$(\lambda_i - \sum_{j=1..H} x_{ij}\mu_{ij})c_i t_i$ – стоимость решения не мигрировавших задач на подсистеме i.

Стоимость обслуживания задач на подсистеме *i*:

$$(\lambda_i - \sum_{j=1..H} x_{ij}\mu_{ij})c_i t_i + \sum_{j=1..H} x_{ij}\mu_{ij}c_j(t_i + kt_i)$$

t_i – среднее время решения задачи i-й подсистемы,

kt_i – штраф за миграцию задачи.

Формирование субоптимальных локальных окрестностей

$$\sum_{i=1..H}((\lambda_i - \sum_{j=1..H} x_{ij}\mu_{ij})c_i t_i + \sum_{j=1..H} x_{ij}\mu_{ij} c_j (t_i + kt_i)) \to \min_{x_{ij}}$$

при ограничениях:

$$x_{ij} = \{0, 1\}$$

$$\sum_{j=1..H} x_{ij} \le l, \quad i = 1,2,\ldots,H$$

Предложен эвристический алгоритм на основе метода цепей Монте-Карло

l – максимально допустимый размер локальной окрестности
H – суммарное количество подсистем

Метод цепей Монте-Карло

АЛГОРИТМ LN_MCC(λ, μ, c, t**)**

```
1   x* ← init()
2   do
3       d ← d0
4       while x* not found and d > dmin do
5           for i ← 0 to imax do
6               x ← gen_sol(x*, d)              /* сгенерировать решение */
7               if f(x, λ, μ, c, t) < f(x*, λ, μ, c, t) then
8                   x* ← x
9                   break
10          end for
11          d ← d / 2                           /* уменьшить "расстояние" */
12      end while
13  while new x* has been found
```

Вероятностный подход

1. При поступлении задачи в ЭМ i ($i \in$ J), выполняется процедура случайного выбора дальнейшей передачи задачи в соответствии с распределением вероятностей $\{\varphi_{ij}\}, j \in J$.
2. При выборе направления j, то задание переходит к этапу формирования подсистемы, а ЭМ j становится корневой для вновь формируемой подсистемы.

Вероятностный подход

“–”

1. Большое время поиска корневой ЭМ, большое среднее время исполнение задания.
2. Большое число шагов до прибытия к корневой ЭМ, что увеличивает нагрузку на сеть ВС,

“+”

1. Простота, точность реализации оптимального плана.
2. Устойчивость к колебаниям внешней нагрузки.

Релаксационный алгоритм

1. Направление (ЭМ i) дальнейшей передачи поступающего в ЭМ j задания определяется из условия $\varphi_{ij} = \max\{\varphi_{ik}\}$, $k \in L(i)$.
2. ЭМ j становится корневой ЭМ для формирования подсистемы.

Релаксационный алгоритм

“–”

Меньшая устойчивость, большая неравномерность нагрузки при распределении заданий.

“+”

Меньшее время поиска корневой ЭМ по сравнению с вероятностными.

Динамический децентрализованный алгоритм распределения заданий

Цель динамического децентрализованного алгоритма распределения заданий – минимизаций заданной функции качества распределения *W*, характеризующийся, в частности, неравномерностью загрузки машин системы при ограничениях на общее число заданий в системе и в каждой ЭМ.

Обозначим $L(i)$ *локальную окрестность* ЭМ i с радиусом ρ, $0 \le \rho \le d$, d – диаметр графа межмашинных связей ВС (множество всех машин, расположенных на расстоянии, не превышающем ρ от ЭМ i).

Динамический децентрализованный алгоритм распределения заданий

1. На основе сбора информации из окрестности $L(i)$ в каждой ЭМ i происходит проверка условий оптимальности текущего вектора распределения заданий.
2. В случае невыполнения условий оптимальности происходит коррекция вектора распределения заданий X. а именно: происходит перемещение заданий в окрестности $L(i)$ с целью минимизации функции W.

Процедура повторяется каждый раз при изменении вектора распределения заданий, постоянно корректируя его с целью оптимизации.

Выбор целевой функции

Пусть целевая функция характеризует неравномерность загрузки.

$$W_1 = \sum_{i=1}^{N} \omega_i = \sum_{i=1}^{N} |xi - Mx|$$

В качестве оценки средней нагрузки по системе Mx в ЭМ i можно использовать величину средней нагрузки ЭМ в окрестности некоторого радиуса относительно ЭМ i:

$$\left(\sum_{k \in L(i)} x_k \right) / |L(i)|$$

Выбор целевой функции

Если известно время t_i выполнения задания q_i. В качестве обобщённой нагрузки ЭМ k может быть взята величина

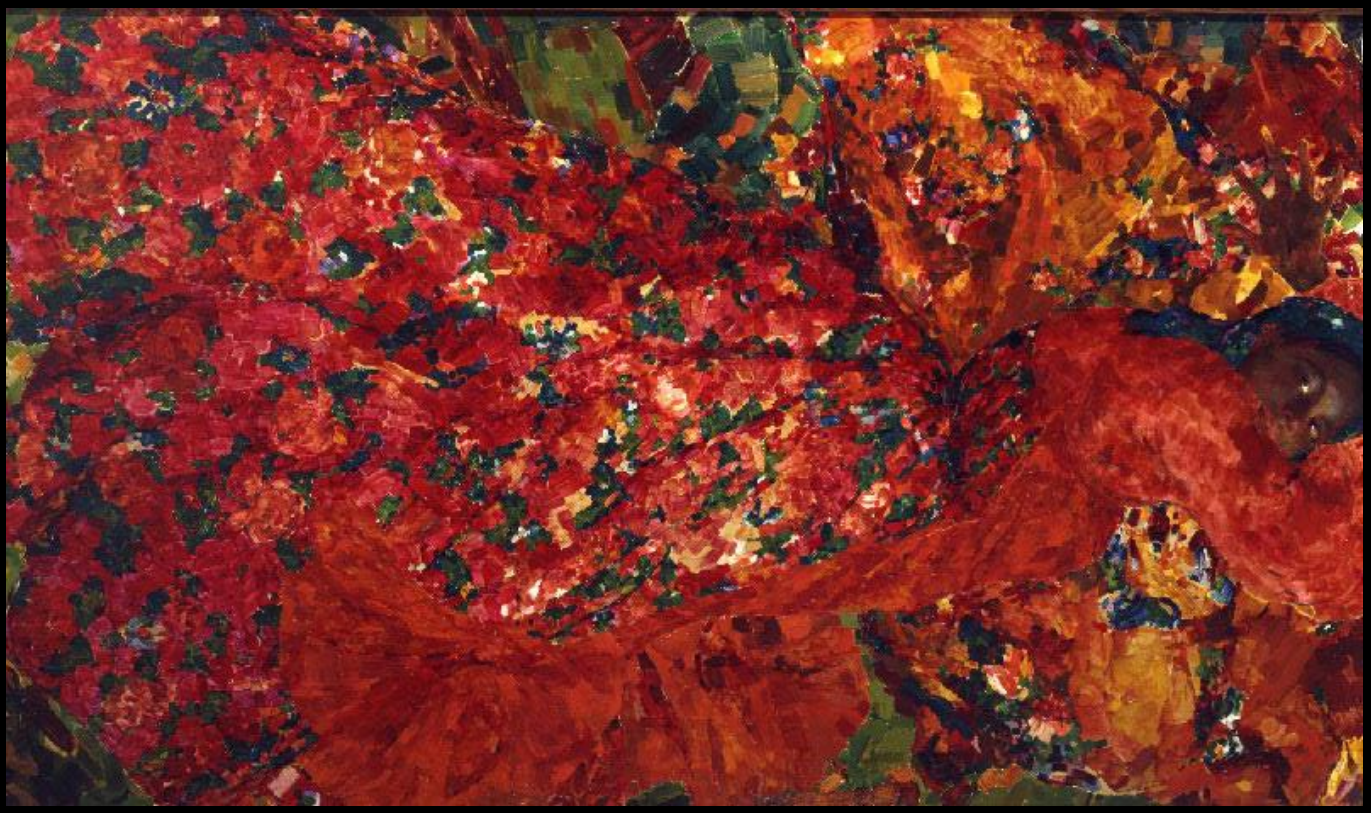
$$h_{ik} = \sum_{k=1}^{x_k} |tj + tixik|$$

Показатель неравномерности загрузки тогда

$$W_i = \sum_{k=1}^{N} |hik(x_{ik}) - Mhk|$$

$$Mh_k = \left(\sum_{j \in L(k)} \sum_{n=1}^{x_j} t_n \right) / |L(k)|$$

$$\sum_{k=1}^{N} x_{ik} = 1$$



Малявин «Танец»