Балансировка

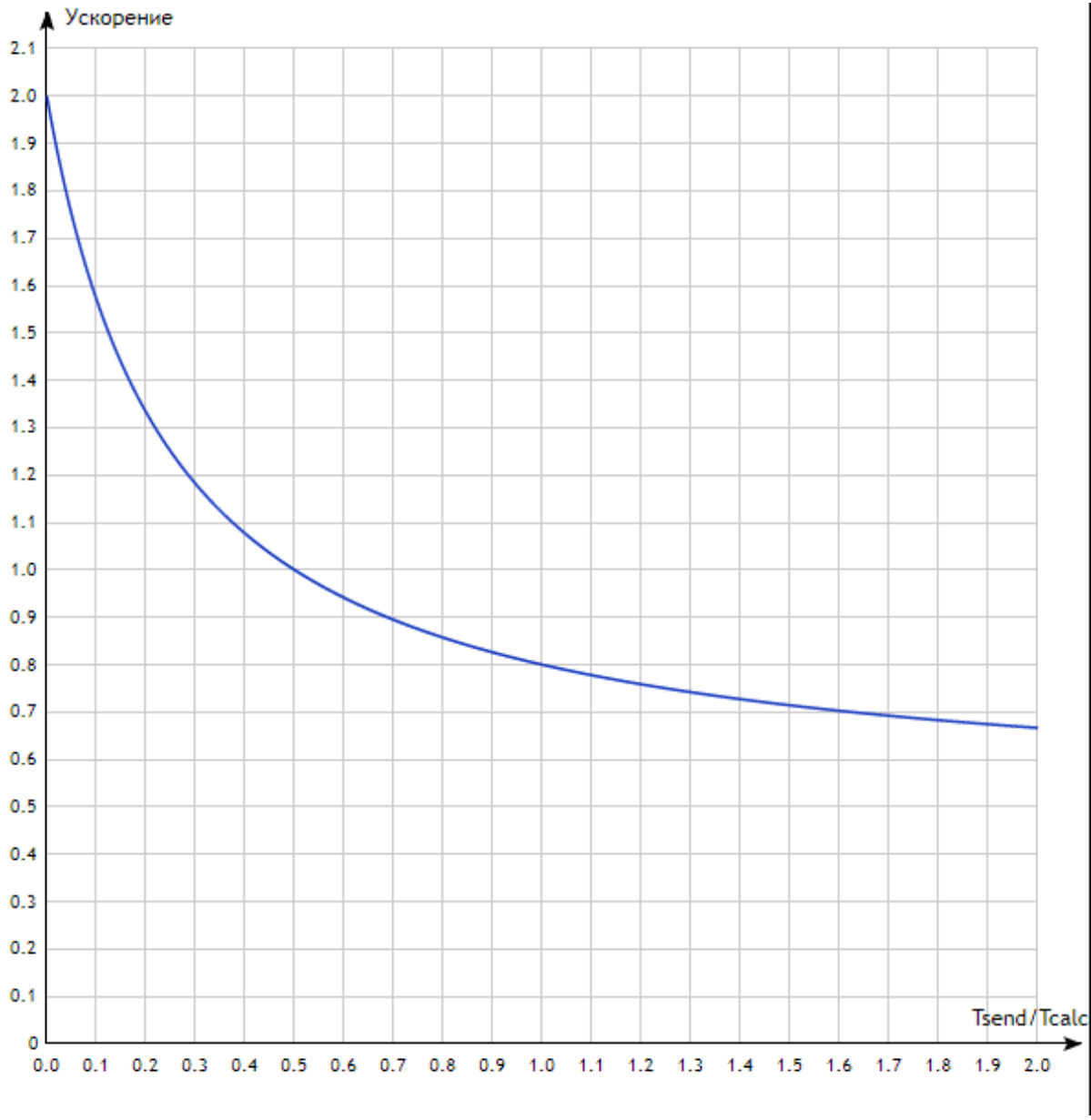
Лисицын Сергей
ФРКТ МФТИ 2020 г.

Замедление ускорения

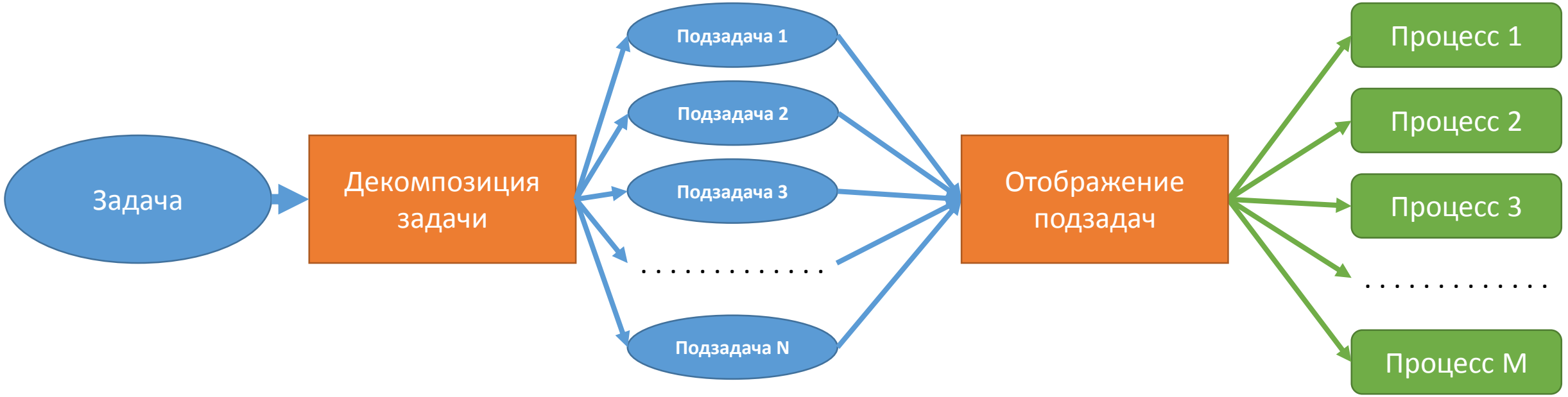
$$T_p = t_{calc} + t_{send}$$

$$T_{2p} = 0.5 * t_{calc} + 2 * t_{send}$$

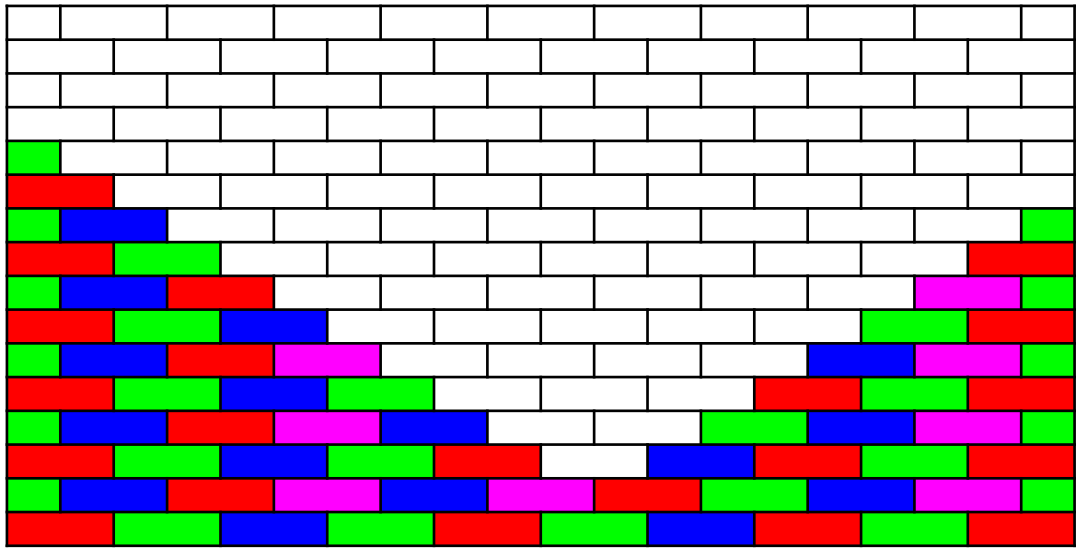
$$S = \frac{T_p}{T_{2p}}$$



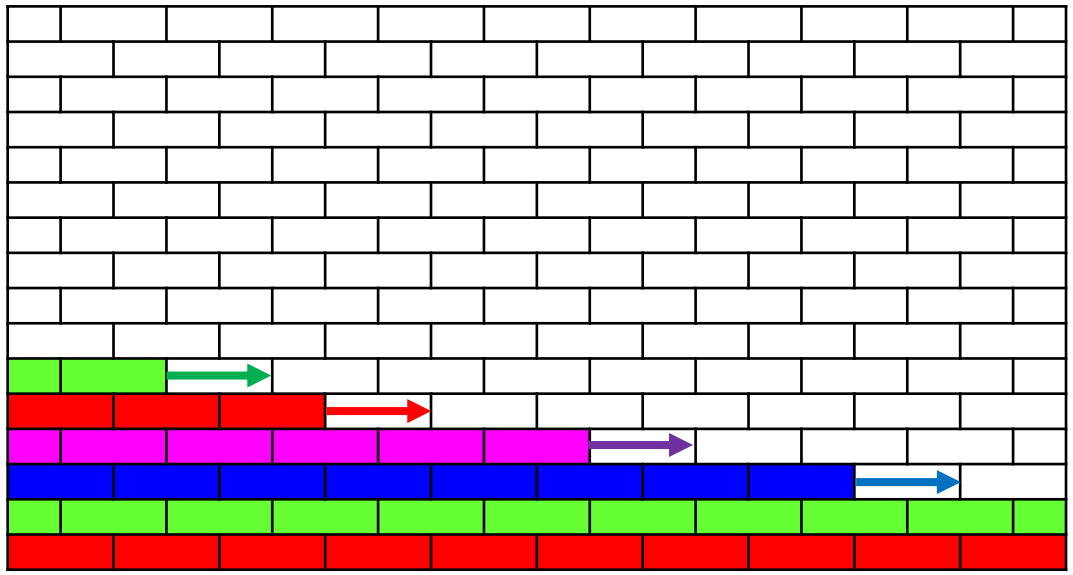
Балансировка нагрузки

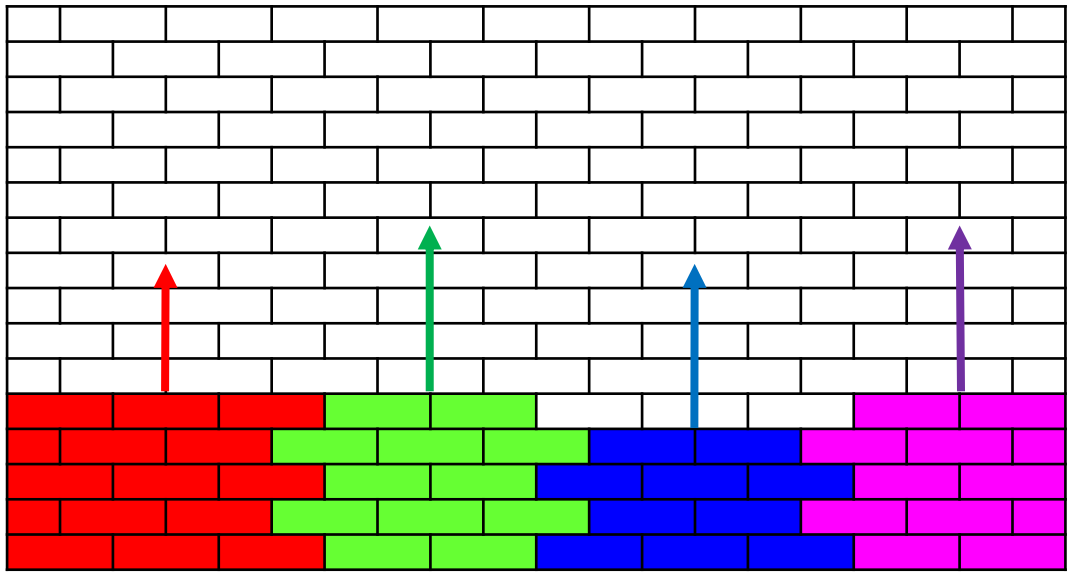


Балансировка нагрузки



Стена Фокса (пример с лекций М.В. Якобовского)

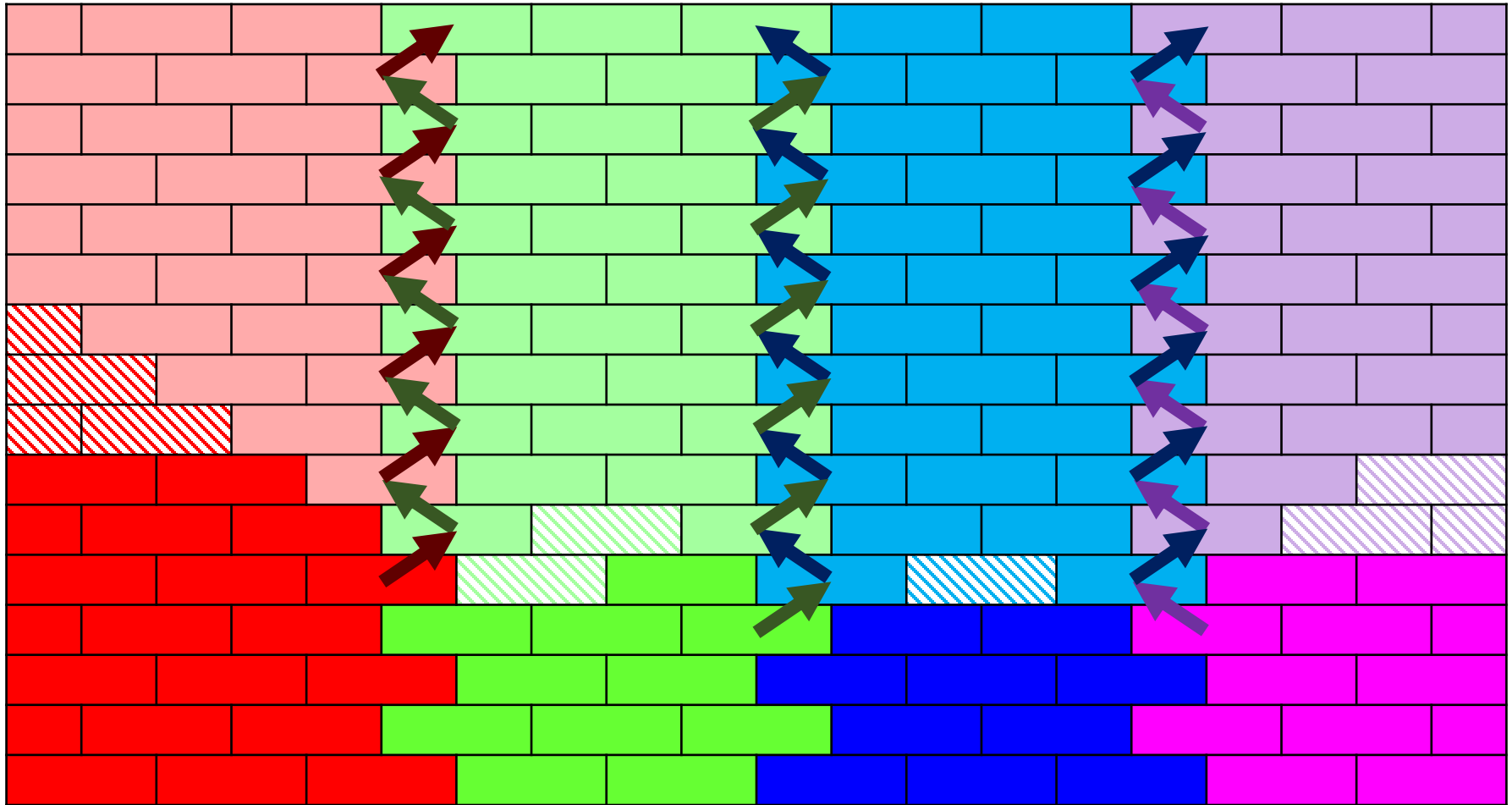




Балансировка нагрузки

- поставленный кирпич
- можно поставить без взаимодействия
- нельзя поставить без взаимодействия
- взаимодействие

Статическая балансировка



Балансировка нагрузки

Проблемы балансировки вычислительной нагрузки:

- структура распределенной задачи неоднородна
- структура вычислительного комплекса (например, кластера) неоднородна
- структура межузлового взаимодействия неоднородна

Балансировка нагрузки

Статическая балансировка

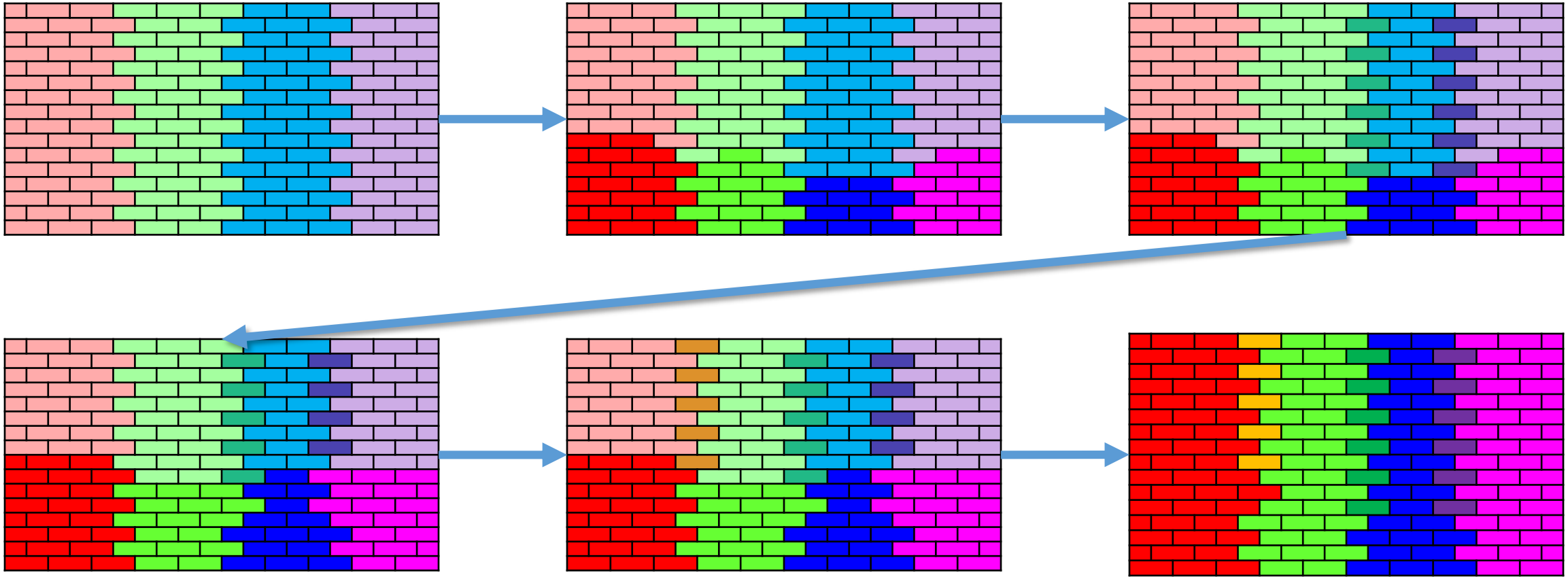
- отображение задач до начала выполнения задачи
- борьба с неоднородностями при помощи эвристик и опыта предыдущих запусков

Динамическая балансировка

- отображение задач происходит до и во время выполнения задачи
- борьба с неоднородностями при помощи постоянного (пере)распределения задач

Балансировка нагрузки

Динамическая балансировка



Балансировка нагрузки

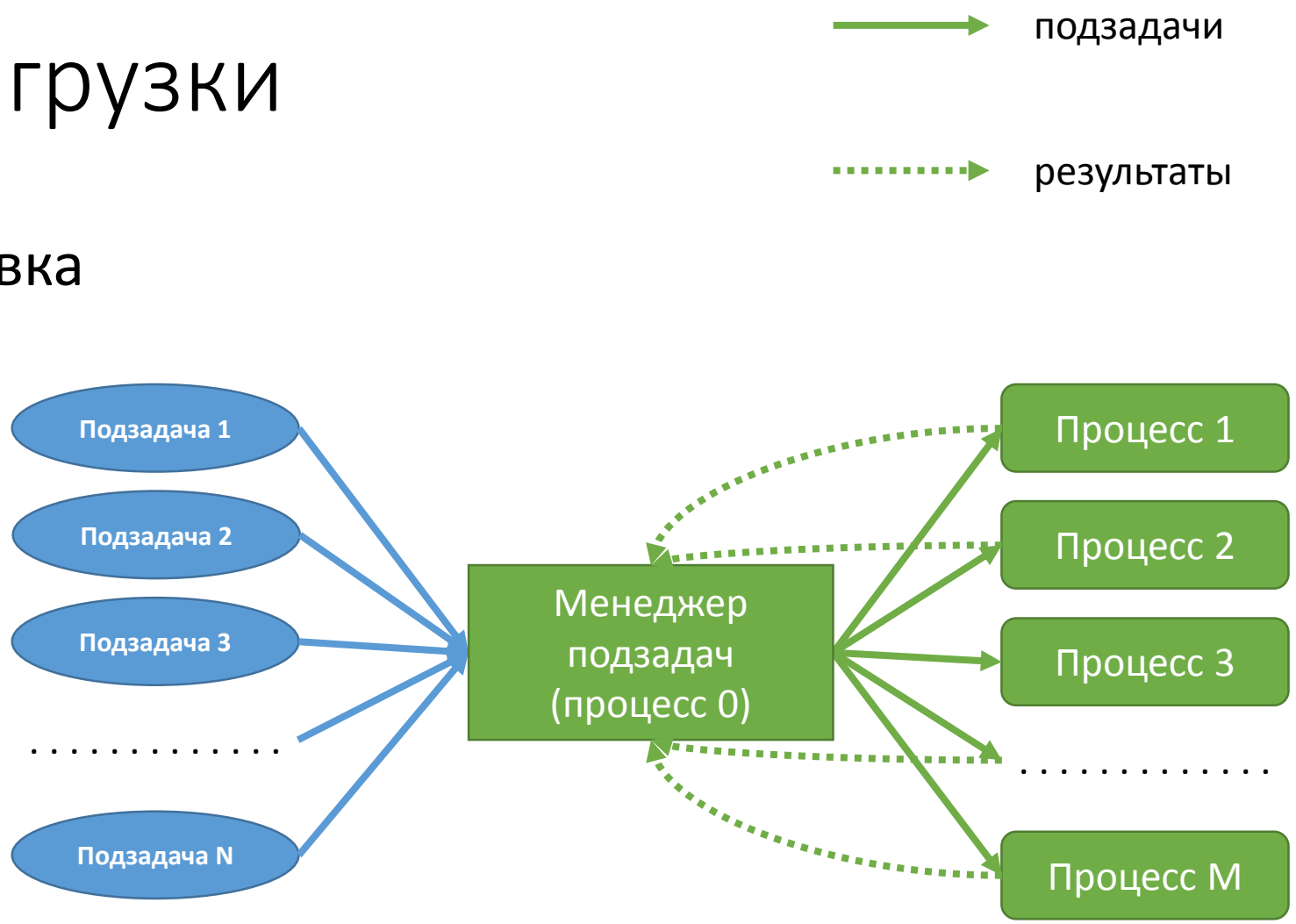
Динамическая балансировка

RCL – стратегия переноса нагрузки:

•случайный алгоритм (random, R)

•алгоритм, основанный на коммуникациях (communication, C)

•алгоритм, основанный на вычислении нагрузки (load, L)



Балансировка нагрузки

Длинная арифметика (сложение)

+	**21**	**43**	**76**	**54**
	4	55	24	02
			0←	56
		1←	100	
	0←	99		
	25	99	00	56

Балансировка нагрузки

Спекулятивные вычисления

	21	43	76	54	53	09	12	94	11	23	08	05
+	4	55	24	45	85	75	25	41	54	25	08	97
			0←	99			1←	135			1←	102
			1←	100			1←	136			1←	102
		1←	100			0←	38			0←	17	
		1←	101			0←	38			0←	17	
	0←	99			0←	84			0←	48		
	0←	99			0←	84			0←	48		
	25				138				65			
	25				138				65			
	25	99	00	99	38	84	38	35	65	48	17	02
	25	99	01	00	38	84	38	36	65	48	17	02

Балансировка нагрузки

Спекулятивные вычисления

+	**21**	**43**	**76**	**54**	**53**	**09**	**12**	**94**	**11**	**23**	**08**	**05**
	4	55	24	45	85	75	25	41	54	25	08	97
			0← 99				1← 135				1← 102	
			1← 100				1← 136				1← 102	
		1← 100				0← 38				0← 17		
		1← 101				0← 38				0← 17		
	0← 99				0← 84				0← 48			
	0← 99				0← 84				0← 48			
	25				138				65			
	25				138				65			
	25	99	00	99	38	84	38	35	65	48	17	02
	25	99	01	00	38	84	38	36	65	48	17	02

Балансировка нагрузки

Суммирование старших разрядов

	21	43	76	54	53	09	12	94	11	23	08	05
+	4	55	24	45	85	75	25	41	54	25	08	97
			0← 99		← 138		1← 135		← 65		1← 102	
			1← 100		139		1← 136		66		1← 102	
		1← 100				0← 38				0← 17		
		1← 101				0← 38				0← 17		
	0← 99				0← 84				0← 48			
	0← 99				0← 84				0← 48			
	25				138				65			
	25				138				65			
	25	99	00	99	38	84	38	35	65	48	17	02
	25	99	01	00	38	84	38	36	65	48	17	02

Балансировка нагрузки

Суммирование старших разрядов

	21	43	76	54	14	09	12	94	11	23	08	05
+	4	55	24	45	85	75	25	41	54	25	08	97
			0←	99	99		1←	135	65		1←	102
			1←	100	100		1←	136	66		1←	102
		1←	100			0←	38			0←	17	
		1←	101			0←	38			0←	17	
	0←	99			0←	84			0←	48		
	0←	99			0←	84			0←	48		
	25				99				65			
	25				100				65			
	25	99	00	99	99	84	38	35	65	48	17	02
	25	99	01	00	99	84	38	36	65	48	17	02

Балансировка нагрузки

- 1000 цифр
- 10 процессов

$$T_1 = 1000 * t$$
$$T_{10} = 100 * 2t + 9T_{sel} + T^*_{gether}$$
$$T^*_{10} = 101 * t + T_{sel} + T_{gether}$$
$$T^{**}_{10} = 100.01 * t + T_{sel} + T_{gether}$$

Балансировка нагрузки

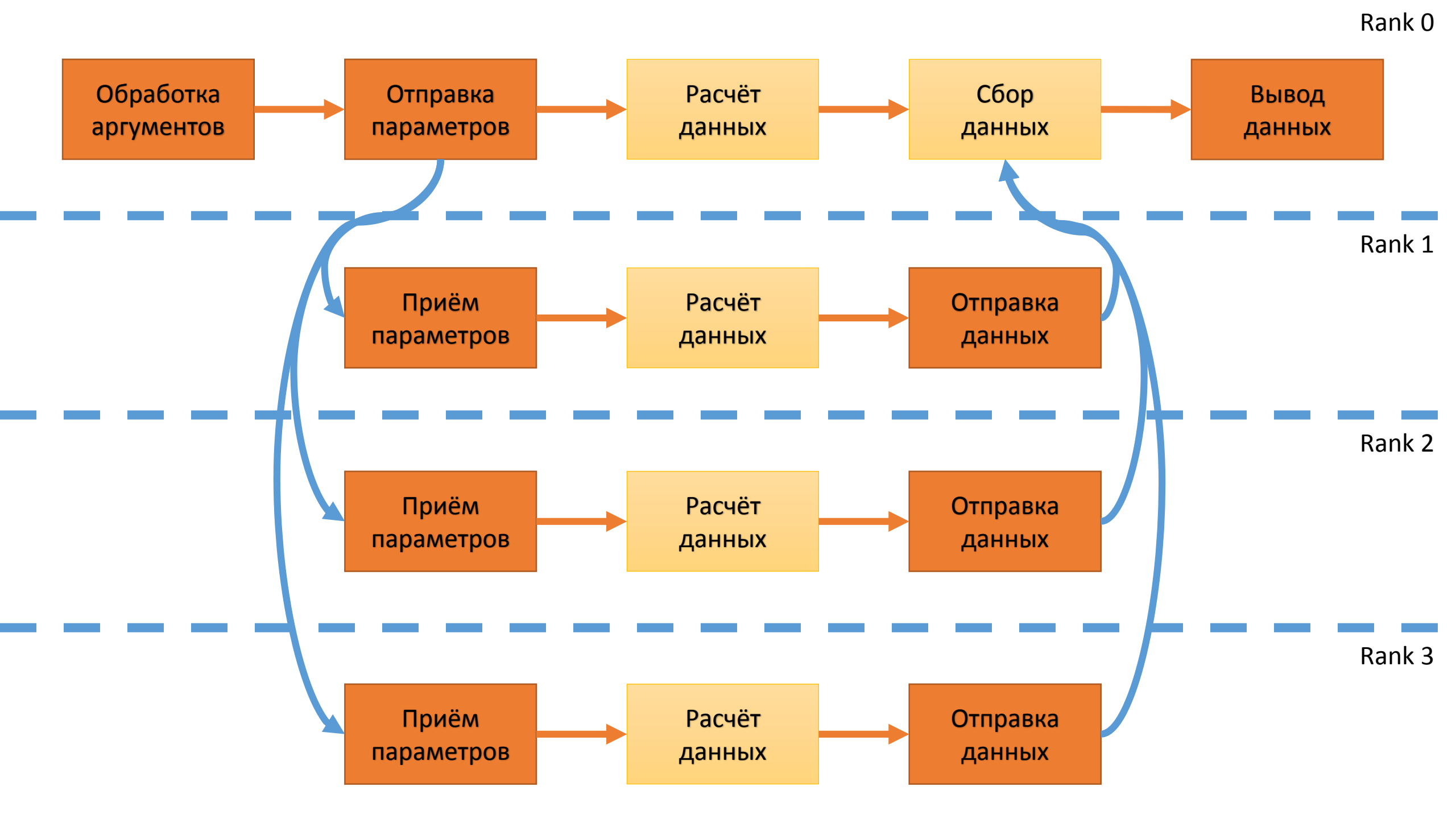
Динамическая балансировка

Исполнитель, t_0							3 …	2 …	1 …
Исполнитель, t_1						3 …	3 +	2 …	1 …
Исполнитель, t_2					1 …	3 …	3 +	2 …	1 +
Первое число	32 разряда	32 разряда	32 разряда	32 разряда	32 разряда	32 разряда	32 разряда	32 разряда	24 разряда
Второе число	32 разряда	32 разряда	32 разряда	32 разряда	32 разряда	32 разряда	32 разряда	32 разряда	24 разряда
Результат (0)							32 разряда		24 разряда
Результат (1)							32 разряда		24 разряда
Перенос разряда (0)							1 разряд		1 разряд
Перенос разряда (1)							1 разряд		1 разряд

Длинное сложение

$ mpirun –n [processes] ./add [input file] [output file]

- Во входном файле 3 строки (длина чисел, первое число, второе число)
- Root-ом считать файл, разбить числа
- Разослать, посчитать локально, перенести разряды, собрать в root-е
- Организовать выдачу блоков для подсчёта, посчитать блоки локально, собрать в root-е, перенести разряды
- В выходном файле root-ом записать результат суммирования
- Вывести время работы алгоритма


Rank 0
Обработка аргументов
Отправка параметров
Расчёт данных
Сбор данных
Вывод данных
Rank 1
Приём параметров
Расчёт данных
Отправка данных
Rank 2
Приём параметров
Расчёт данных
Отправка данных
Rank 3
Приём параметров
Расчёт данных
Отправка данных

Длинное сложение

Input:

18

568432054987453034

984532106480023443

Output:

155296416146747 6477

Ограничения:

- Количество процессов 2^k+1
- Количество цифр 9*2^m, (m >= k)