Runtime, memory & scalability

Сколько стоит локальное происхождение: время, память, масштаб

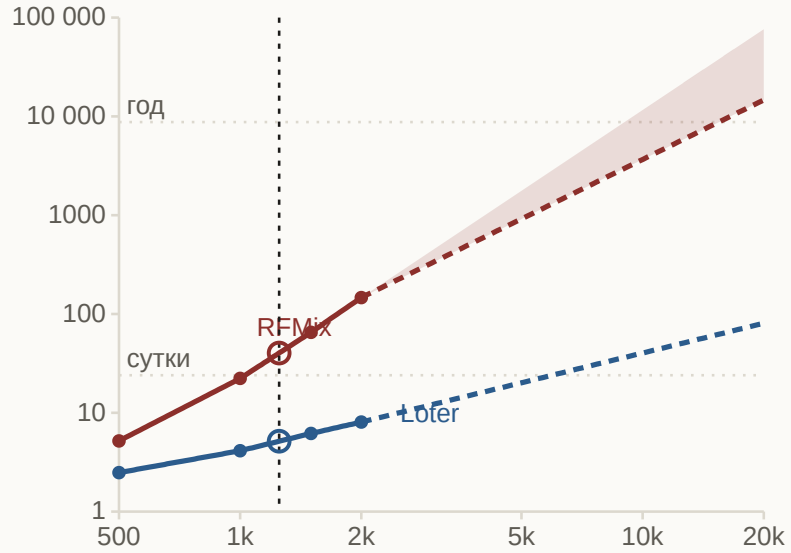
Максим Кирилюк · НИС АДНЖ · 19.11.2026

Данные: Schubert, Andaleon & Wheeler (2020), *PeerJ* 8:e10090

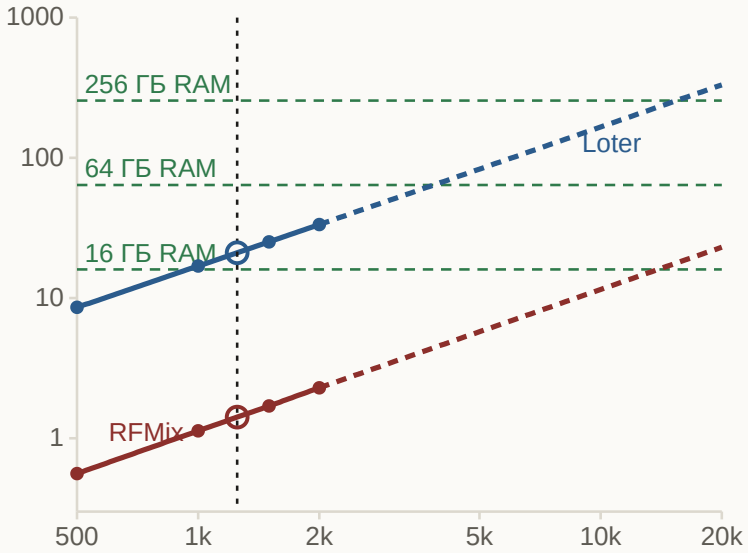
2000 человек, одно ядро: 8 часов или 6 суток?

людей в когорте — **N = 1 250**

время на одном ядре, ч (лог. оси)



пик памяти, ГБ (лог. оси)



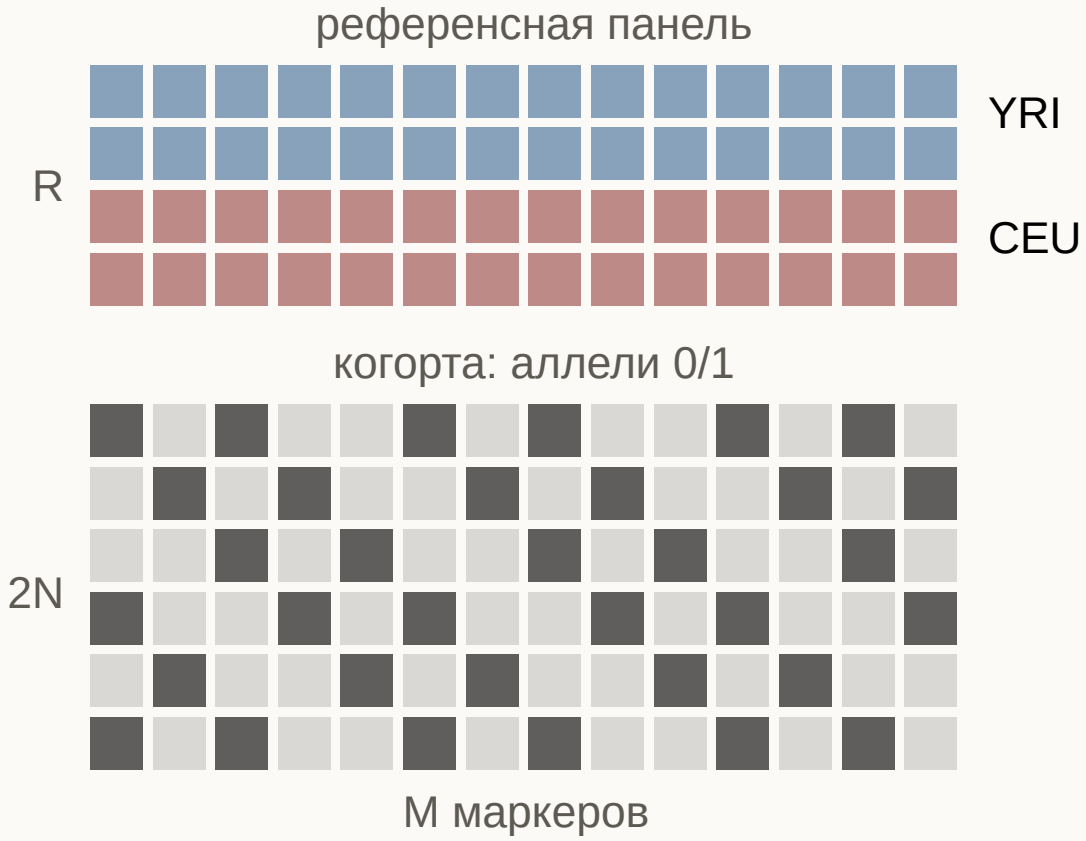
Между замерами. RFMix: 40 ч, память 1,4 ГБ — влезет в 16 ГБ.

Loter: 5,2 ч, память 21 ГБ — влезет в 64 ГБ.

Время: Loter растёт как N, RFMix — быстрее N^2. Память — наоборот: у Loter в 15 раз больше.

Точки — замер (Schubert 2020, Fig. 2–3, N = 500…2000, одно ядро, M ≈ 50 000 SNP). Дальше — экстраполяция: RFMix между α = 2 и α = 2,7 (полоса), Loter — прямая.

Размер задачи — четыре числа



N	люди (2N гаплотипов)	20…2000
M	маркеры (SNP)	50 000*
R	гаплотипы панели	≈ 372
K	популяции	2

Меняли только **N**.

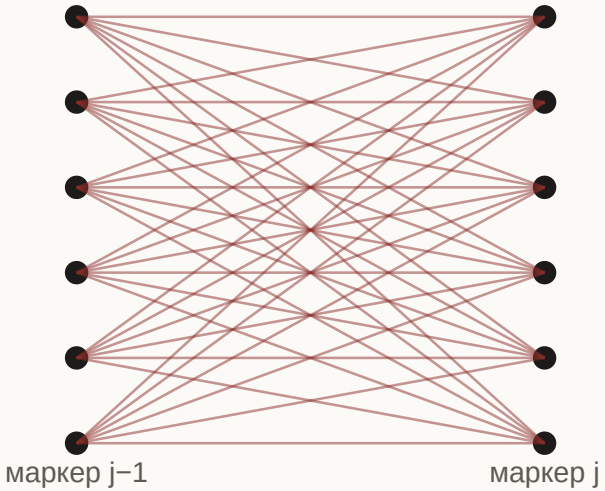
Вход при N = 2000: $2 \cdot 10^8$ аллелей = **25 МБ** битами.

* M для этого опыта в статье не назван: 50 000 (хр. 22) или 44 332 (хр. 1); берём 50 000 (±10%). R: 1000G CEU 99 + YRI 108 минус ≈10% основателей — оценка.

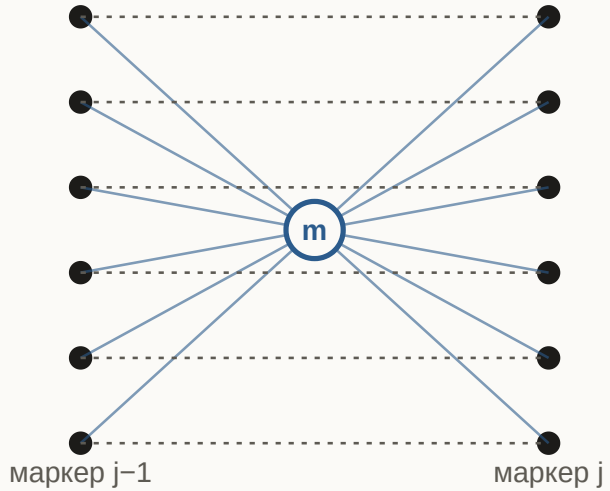
I. Время: один гаплотип — динамика за O(M·R)

гаплотипов в панели — **R = 6**

в лоб: каждая строка смотрит на все — **36 на маркер**



с общим минимумом — **12 на маркер**



$c_j(l) = [h_j \neq A_{lj}] + \min(c_{j-1}(l), m + \lambda),\ m = \min_{l'} c_{j-1}(l')$

Один гаплотип, M = 50 000 маркеров: в лоб $1{,}8 \cdot 10^6$, с минимумом $6{,}0 \cdot 10^5$ — в **3** раза меньше.

Штраф λ за прыжок одинаковый для всех строк, поэтому из всех прыжков в строку l выгоден только прыжок из минимума прошлого столбца m. Это видно в Loter: `src/graph/graph.cpp`, `EstimateSknn::s_cost_i`. В HMM (Li & Stephens) то же самое с суммой вместо минимума.

Loter: 160 проходов, и во что они упираются

$$T \propto 8_{\lambda} \times 20_{\text{бутстрэп}} \times 2N \times M \times R$$

N = 2000: **1,2·10¹³** клеток.

Соседи по когорте не участвуют ⇒ **линейно по N**.

Таблица на гаплотип: R × M × 8 Б = **149 МБ** — ни в один кэш.

Если упираемся в запись в DRAM:

95 ТБ / 10–20 ГБ/с ≈ **1,3–2,6 ч**; замер — **8,1 ч**. Тот же порядок.



↑ таблица DP 149 МБ живёт здесь

Типичные порядки для x86-сервера; машина в статье не указана. Оценка, не профиль. Ещё одно слагаемое того же порядка — перекраска на Python: цикл по R номерам панели (local_ancestry.py, clusters_to_list_pop).

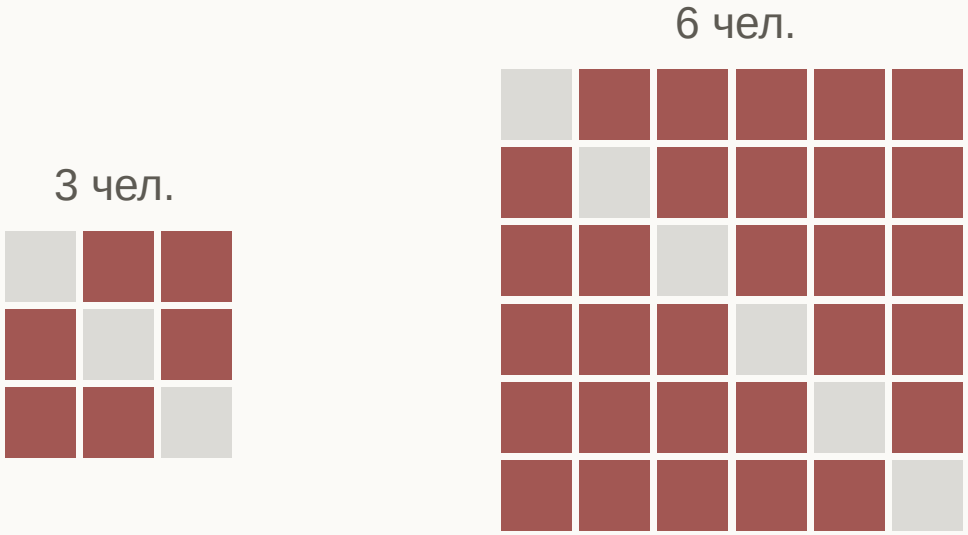
RFMix: откуда N²

В каждом окне — случайный лес, обученный на панели: это линейно.

EM-доучивание (`-e 2`): каждому человеку — свой лес на *остальной когорте*.

N лесов × 2N примеров ≈ 2N²

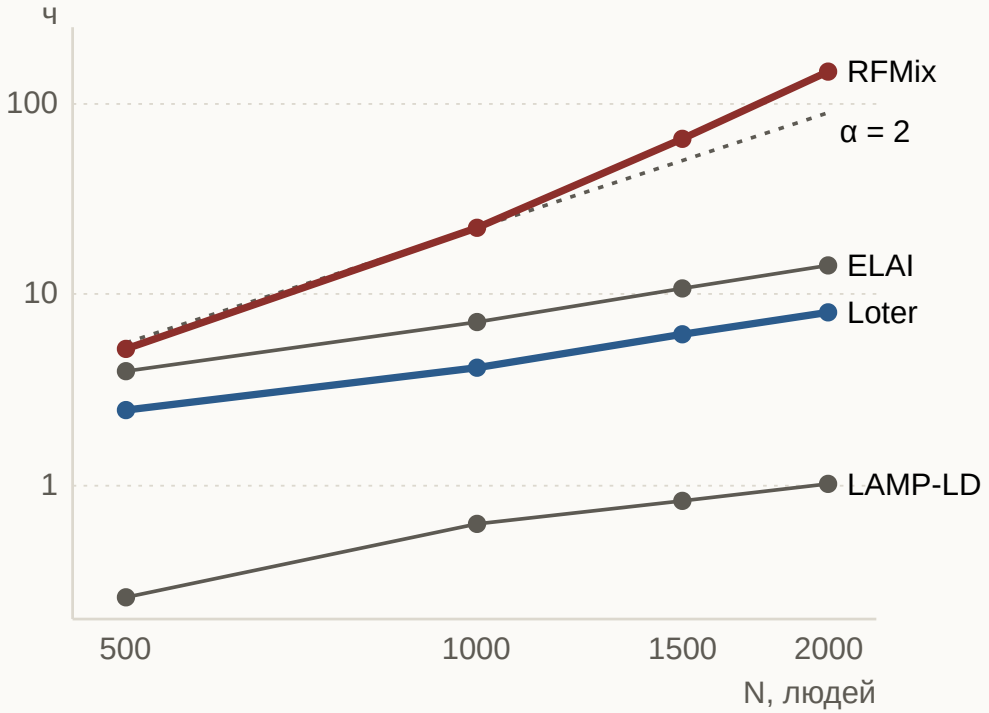
Людей ×2 ⇒ лесов ×2, каждый ×2 ⇒ **работа ×4**.



строка — лес одного человека,
клетки — его обучающие примеры

По коду RFMix 1.5.4 (main.cpp, randomforest.cpp); запуск авторов без --use-reference-panels-in-EM. В статье механизма нет.

Замер: наклон в log–log



$$\alpha = \log(T_2/T_1) / \log(N_2/N_1)$$

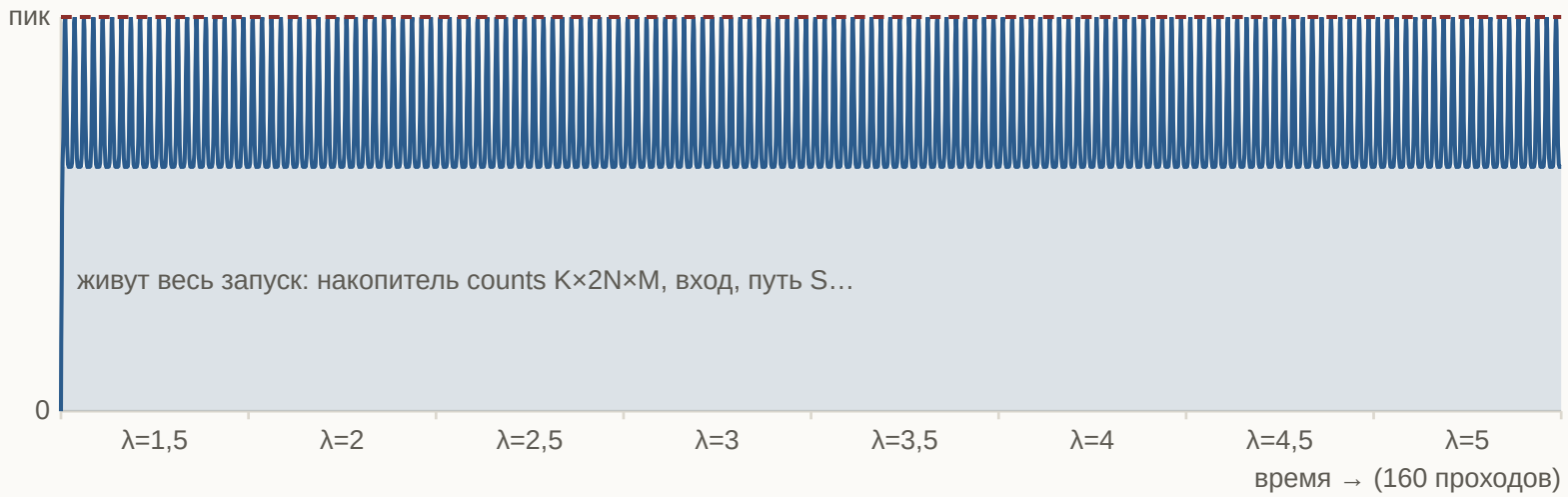
	500 → 1000	1000 → 2000
Loter	0,74	**0,96**
ELAI	0,86	0,98
RFMix	2,10	**2,72**

$T = a + bN + cN^2$, $a,b,c \geq 0 \Rightarrow T(2N) \leq 4T(N) \Rightarrow \alpha \leq 2$.
RFMix: ×6,57 ⇒ α = 2,72. Даже $N^2 \cdot \log(2N)$ даёт ≈ 2,1.
Лишний рост не объяснён.

Точки сняты с Fig. 2 (Schubert 2020), одна точка на размер; N < 500 сливаются. Пунктир — α = 2.

II. Память: пик, а не сумма

проход ▶ | +20 | сначала | проход 160 / 160

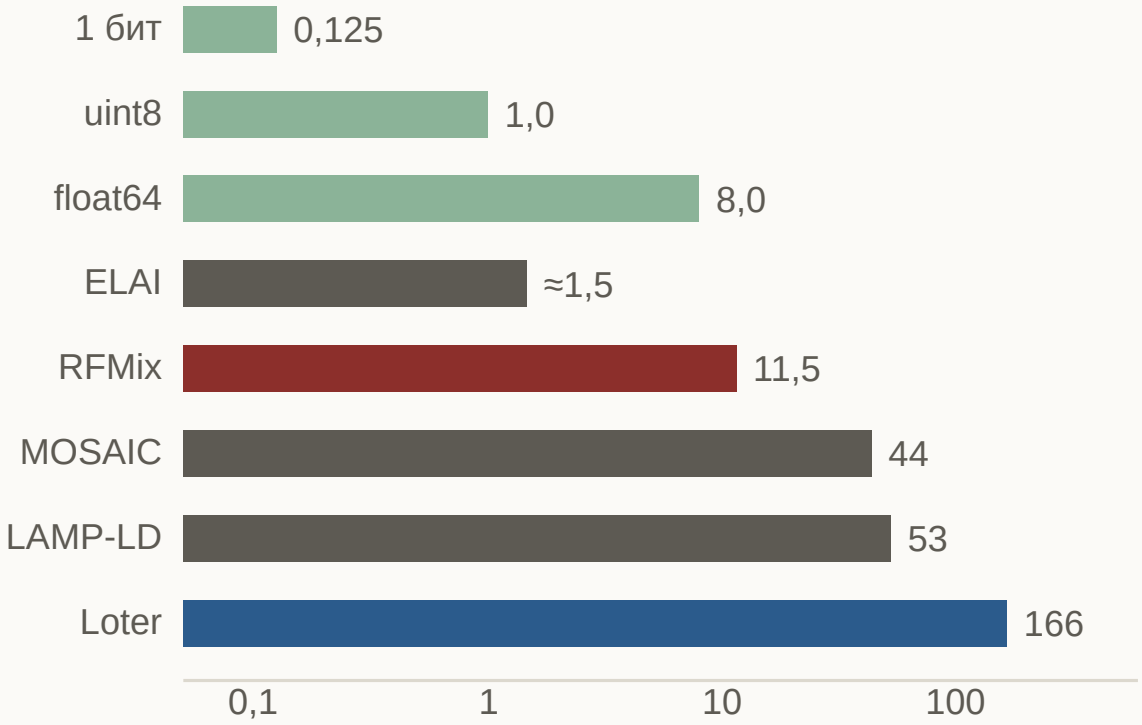


время (сумма)		8,1 ч
память (пик)		33,4 ГБ

Итог: время **×160**, память **×1**. С N растёт только то, что живёт весь запуск: массивы 2N×M.

Loter, N = 2000, одно ядро: 8 значений λ × 20 бутстрэп-панелей = 160 проходов по очереди. Профиль памяти — схема; числа — замер (Schubert 2020, Fig. 2–3). `time -v` сообщает пик: Maximum resident set size.

Сколько байт на один аллель



$$s = (\Delta M_{peak}/\Delta N) / 2M$$

Информации в аллеле — **1 бит**.

Loter хранит **≈ 166 Б**:

вход 25 МБ → **33 ГБ**.

Память задаёт *представление* данных, а не их объём.

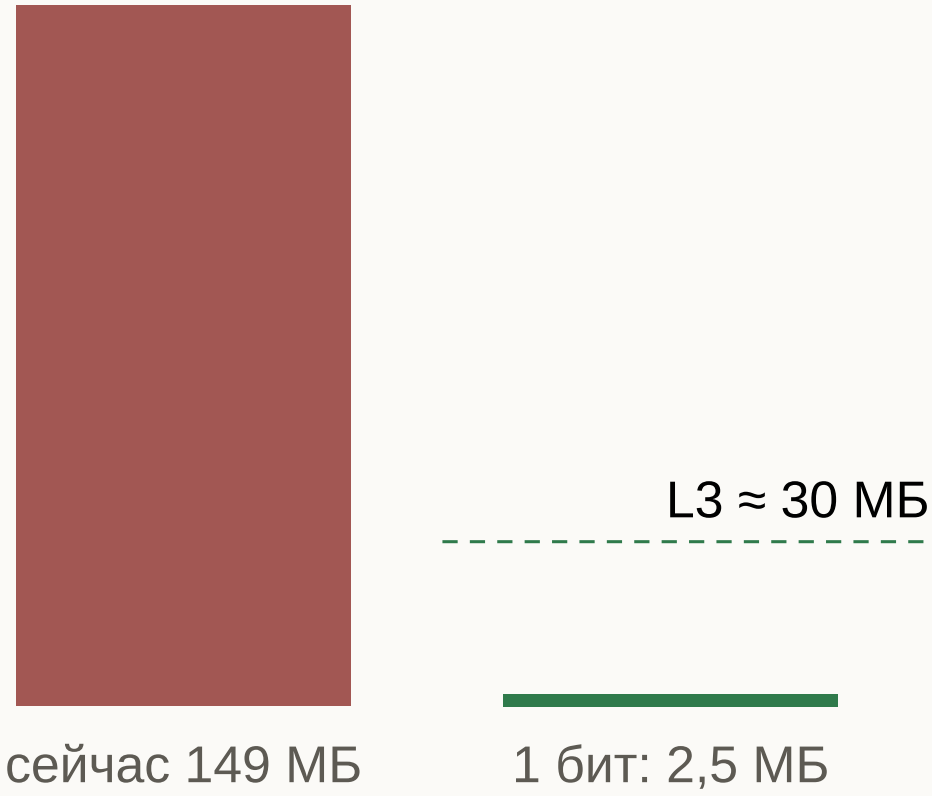
Наклон по N = 500 и 2000 на Fig. 3; 2M = 10^5 аллелей на человека. По коду Loter явные массивы 2N×M дают ≥ 49 Б; остальное не профилировано.

Хранить меньше: 149 МБ → 2,5 МБ

1. стоимости нужен только *предыдущий* столбец
2. путь из клетки ведёт либо в свою строку, либо в **минимум** — это **1 бит**

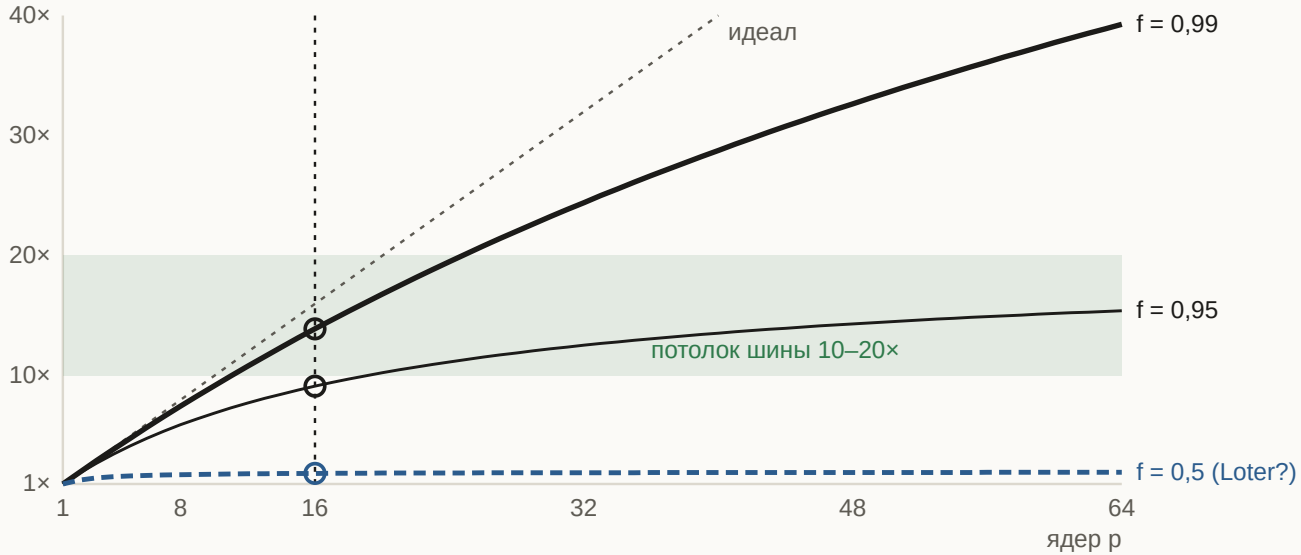
$R \cdot M/8 + 4M \approx 2{,}5$ МБ (×59 меньше)

Операций столько же, но таблица влезает в L3.



III. Масштабирование: пределы ускорения

ядер — **p = 16**



`S(p) = 1 / ((1−f) + f/p) ≤ 1/(1−f)`

На 16 ядрах: f = 0,99 → **13,9×** (предел 100×), f = 0,95 → **9,1×** (предел 20×).

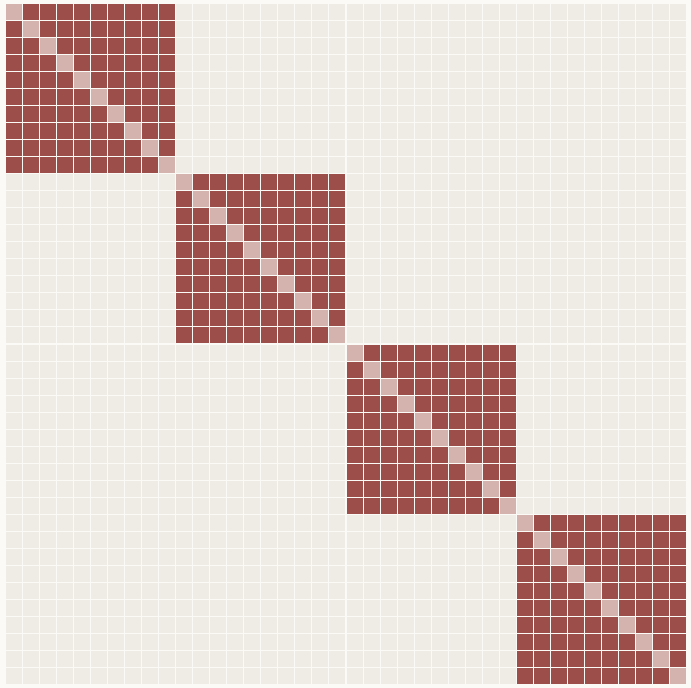
Если проход упирается в память, все ядра делят одну шину: выше ~10–20× не подняться.

Loter: динамика на C++ параллельна (OpenMP по гаплотипам), а перекраска на Python — нет. Если она того же порядка, f ≈ 0,5 и S ≤ 2 (оценка по коду).

Закон Амдала: доля f работы параллельна, остальное идёт последовательно. Потолок шины — типичные порядки (≈100–300 ГБ/с на узел, 10–20 ГБ/с на ядро), не замер. В статье всё считали на одном ядре.

Делим когорту на p частей

частей — **p = 4 (по 500 чел.)**



квадрат N×N: кто на ком учится при EM
цветное — работа, которая осталась

Работа: $p \cdot c(N/p)^2 = cN^2/p \to$ в **4** раз меньше *даже на одном ядре*.
RFMix, одно ядро: замер **20,8 ч** (модель 36,6 ч) вместо 146,6 ч.
На 4 ядрах параллельно: ≈ 5,2 ч.
Цена: EM учится на 998 гаплотипах вместо 3998 — точность надо проверять.

p	модель, ч	замер, ч
1	146,6	146,6
2	73,3	44,6
4	**36,6**	**20,8**

Замер даёт ×7, больше модельных ×4: время RFMix растёт быстрее N^2 ($\alpha \approx 2{,}4$ на 500→2000).

RFMix, N = 2000, одно ядро. «Модель» — $c \cdot N^2/p$ по замеру 146,6 ч. «Замер» — p частей по N/p человек, время N/p взято с Fig. 2 (Schubert 2020); точки N < 500 не снимаются надёжно.

Три числа для любого метода

1. **α** — как растёт время: замер на 3+ размерах, наклон в log–log
2. **Байт на аллель** — сколько людей влезет: $N_{max} \approx RAM / (2M \cdot s)$
3. **Единица деления** — можно ли резать когорту и что станет с точностью

	α	Б/аллель	точность
LAMP-LD	≈ 0,7	53	
ELAI	1,0	≈ 1,5	
Loter	1,0	166	ниже RFMix
MOSAIC	—	44	
RFMix	2,7	11,5	лучшая медиана

α на 1000→2000. Не измерено в статье: несколько ядер, другие M и R, повторы, профиль.