USPARS - параллельный прямой решатель СЛАУ

Новосибирский центр информационных технологий
УНИПРО

17 ноября 2022

USPARS – прямой решатель СЛАУ

Предпосылки создания пакета:

- ✓ Идея - разработать конкурентноспособный мультиплатформенный пакет для решения СЛАУ с разреженными матрицами коэффициентов методом Гаусса, доступный, в том числе, на разрабатываемых в России процессорах
- ✓ Не было готового продукта для отечественных CPU (Эльбрус, Байкал)
- ✓ Наличие команды, обладающей экспертизой в разработке прямых решателей и сопутствующих алгоритмов, позволяло подумать о конкурентоспособности не только в России, но и в мире
- ✓ Возможность предоставления быстрой качественной поддержки для российских пользователей
- ✓ Возможность полного контроля над исходным кодом пакета.
- ✓ Опыт работы в т.ч. с зарубежными заказчиками. Опыт показывает, что выгоднее иметь свой собственный продукт, который стоит дешевле имеющихся на рынке и который можно настраивать и расширять под свои нужды
- ✓ Было время и ресурсы на создание пакета из-за ухода «больших» клиентов с 2019 г.

Состав USPARS

Функциональное наполнение - классическое для прямых решателей разреженных матриц:

- ✓ Полный набор функций для осуществления треугольной факторизации и решения СЛАУ: LU, LDLT, LDLH, Холецкий разреженных матриц с вещественными и комплексными элементами одинарной и двойной точности.
- ✓ Собственная компонента (Atlant) переупорядочивания элементов матрицы для уменьшения заполненности факторов и улучшения распараллеливаемости
- ✓ Средства повышения численной устойчивости процесса решения (scaling, boosting и global pivoting)
- ✓ Возможность использования итерационного уточнения для улучшения точности решения

Оптимизация производительности для серверов с общей памятью на основе многоядерных процессоров:

- ✓ Функции USPARS распараллелены средствами библиотеки OpenMP и базируются на использовании общеупотребимых функций высокооптимизированных библиотек (BLAS).

PyUspars: scipy-совместимый python-интерфейс:

- ✓ Простой (близкий к естественному) синтаксис задания запросов на решение
- ✓ Автоматический контроль за выделением, распределением и освобождением динамической памяти
- ✓ Использование стандартного для Python Sparse CSR формата матриц

Python-пакет. Преимущества

Удобный пользовательский интерфейс:

- ❑ Простой (близкий к естественному) синтаксис задания запросов на решение
- ❑ Автоматический контроль за выделением, распределением и освобождением динамической памяти

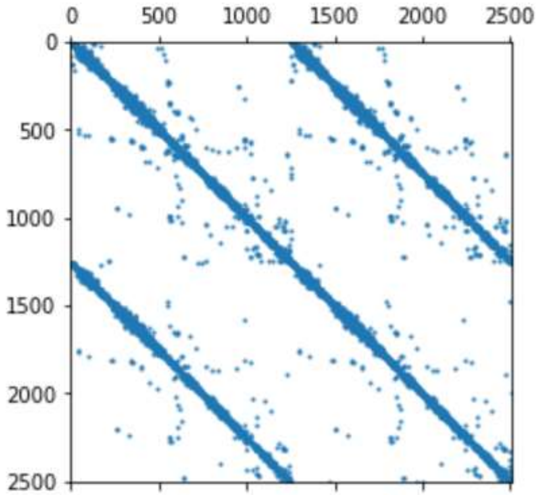
Расширенный функционал, направленный на верификацию полученного решения:

- ❑ Расчет невязки, погрешности
- ❑ Визуализации исходной матрицы, решения, перестановок
- ❑ Нахождение числа обусловленности матрицы

Используется в качестве инструмента для написания и тестирования нового функционала

Пример решения системы с визуализацией матрицы:

```
A = mmread("1.mtx").tocsr()
plt.spy(A, markersize=1)
x_exact = [1]*A.shape[0]
b = A*x_exact
```




```
tt = PyUspars()
tt.init(A, ftype.LU)
tt.fact()
x = tt.solve(b)
print("Error -", tt.calc_norm_error(x_exact))
print("Discrepancy -", tt.calc_norm_residual())
```

```
Error - 1.4934960850244645e-12
Discrepancy - 7.13933003711876e-14
```

Python-пакет. SciPy-совместимость

- Полная совместимость со стандартным для Python Sparse CSR форматом матриц
- Доступны все встроенные в SciPy оптимизированные операции над разреженными матрицами
- Возможность использования совместимых с этим форматом визуализаций, методов работы с данными, чтения и записи
- Обратная совместимость интерфейсов прямых разреженных решателей, позволяющая использовать Uspars вместо стандартных методов SciPy путем замены импорта.

```
from pyuspars_package.solver import PyUspars
from pyuspars_package.solver import FACTORIZE_TYPE as ftype
from pyuspars_package.scipy import factorized, spsolve
import numpy as np
from scipy import sparse
from scipy.io import mmread
import matplotlib.pylab as plt
```

Замена импорта:

```
#from scipy.sparse import spsolve
from pyuspars_package.scipy import spsolve
```

Статус работ

- ✓ Uspars оттестирован в плане точности, надежности и производительности на **наборе тестовых матриц**, взятых с ресурсов https://sparse.tamu.edu/ (**800 матриц**), http://www.elses.jp/matrix/ и матрицы заказчика.
- ✓ Надежность, точность и производительность Uspars не уступают, а во многих случаях и **превосходят показатели** для зарубежных конкурентов.
- ✓ Uspars работает на платформах **x86, Эльбрус, АРМ**.
- ✓ Оптимизирована производительность для серверов с общей памятью на основе **многоядерных процессоров**:
  - ✓ Параллелизация средствами OpenMP;
  - ✓ Использование высоко-оптимизированных математических библиотек.
  - ✓ Ряд авторских алгоритмических оптимизаций.

Краткое описание архитектуры USPARS

$$Ax = b$$
$$A' = D_l P P_m A P^t D_r$$

P − матрица перестановок перестановка, минимизирующая заполняемость ненулевыми элементами треугольных факторов (fill-in)

P_m − перестановка больших элементов A на главную диагональ (matching);

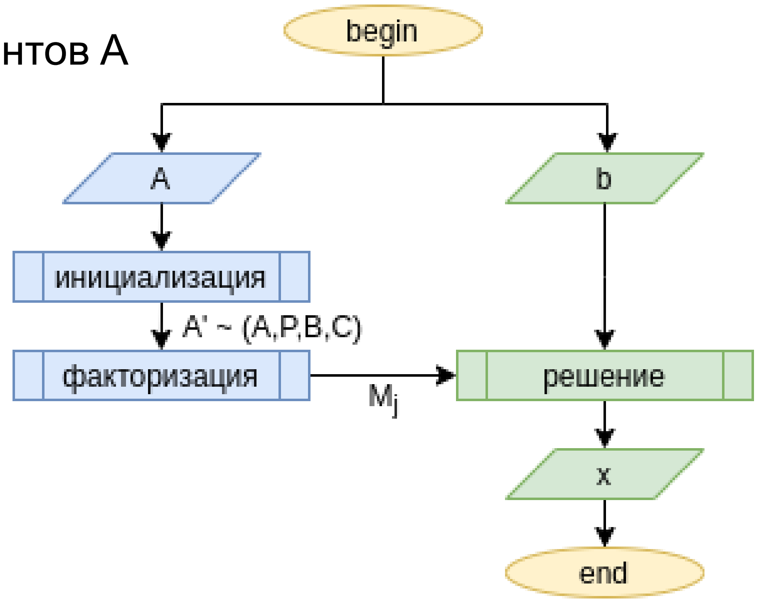
D_l, D_r - диагональные матрицы: масштабирование элементов A (scaling).

$$A' = \{LL^H | LDL^H | LDL^T | LU\},$$

L – нижнетреугольная;

U – верхнетреугольная;

D – диагональная.

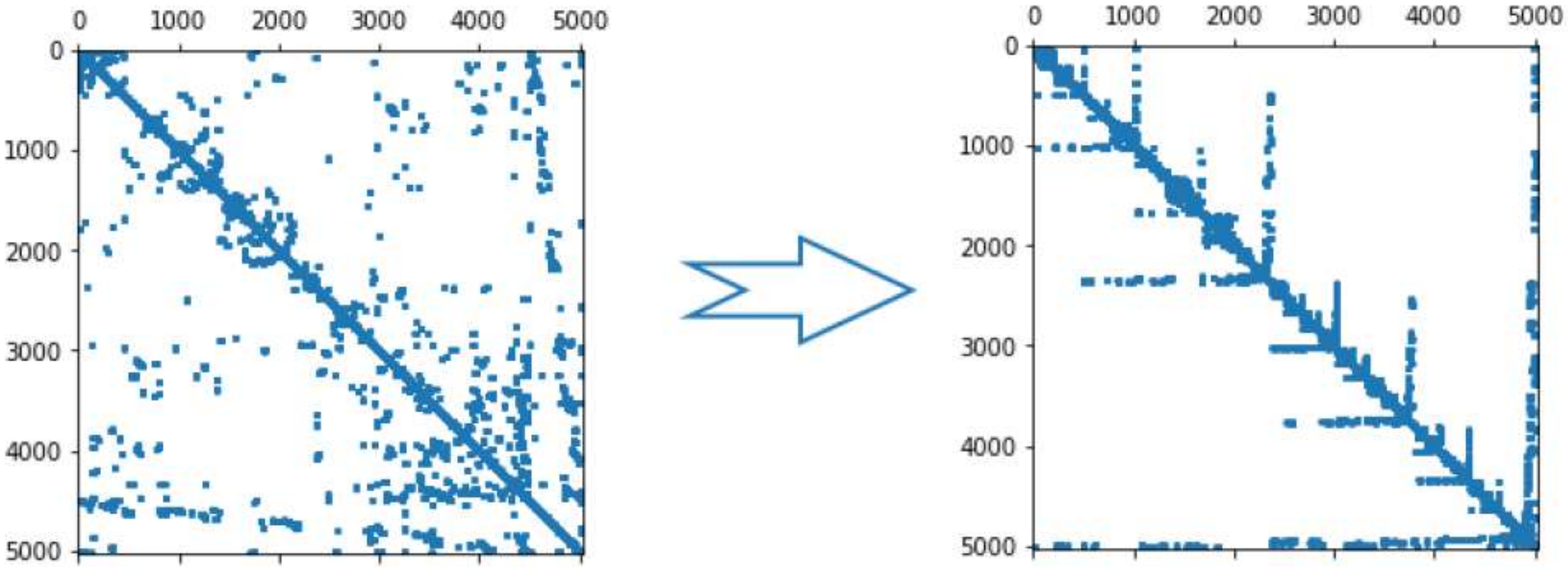


Уменьшения заполненности факторов (fill-in)

$$A = \begin{pmatrix} 9 & 1.5 & 6 & 0.75 & 3 \\ 1.5 & 0.5 & & & \\ 6 & & 12 & & \\ 0.75 & & & 0.625 & \\ 3 & & & & 16 \end{pmatrix} = LL^t \qquad L = \begin{pmatrix} 3 & & & & \\ 0.5 & 0.5 & & & \\ 2 & -2 & 2 & & \\ 0.25 & -0.25 & -0.5 & 0.5 & \\ 1 & -1 & -2 & -3 & 1 \end{pmatrix}$$

$$A' = PAP^t = \begin{pmatrix} 16 & & & & 3 \\ & 0.5 & & & 0.75 \\ & & 12 & & 6 \\ & & & 0.625 & 1.5 \\ 3 & 1.5 & 6 & 0.75 & 9 \end{pmatrix} = LL^t \qquad L = \begin{pmatrix} 4 & & & & \\ & \frac{1}{\sqrt{2}} & & & \\ & & 2\sqrt{3} & & \\ & & & \frac{\sqrt{10}}{4} & \\ \frac{3}{4} & \frac{3}{\sqrt{2}} & \sqrt{3} & \frac{3}{\sqrt{10}} & \sqrt{\frac{3}{80}} \end{pmatrix}$$

Примеры портретов A, L



Результаты сравнительных прогонов: USPARS/PARDISO/MUMPS

Постановка эксперимента:

- ✓ Найти решение СЛАУ с одной правой частью сгенерированной по заданному вектору решений
- ✓ Матрицы:
  - ✓ Suite Sparse Matrix Collection (https://sparse.tamu.edu/)
  - ✓ 24 симметричных и 12 несимметричных
  - ✓ Размер СЛАУ 14 000 … 5 500 000 неизвестных
- ✓ Система:

№1 (AMD, 128 ядер)	№2 (Intel, 24 ядра)	№3 (Эльбрус, 32 ядра)
✓ 2 процессора AMD EPYC 7763 64-Core ✓ Оперативная память 1024 Gb ✓ Aramid 3.1 (Red Hat Enterprise Linux 8.3)	✓ 2 процессора Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz (12 core, HT disabled) ✓ 256 GB RAM ✓ Ubuntu 20.04.3 LTS	✓ 4 процессора E8C-SWTX @ 1.20GHz (8 ядер) (http://ineum.ru/e8c-swtx) ✓ 128 GB RAM ✓ OS Elbrus 7.2

- ✓ Проверяемые характеристики:
  - ✓ Точность решения
  - ✓ Размер используемой памяти
  - ✓ Производительность

USPARS/PARDISO/MUMPS: Список матриц

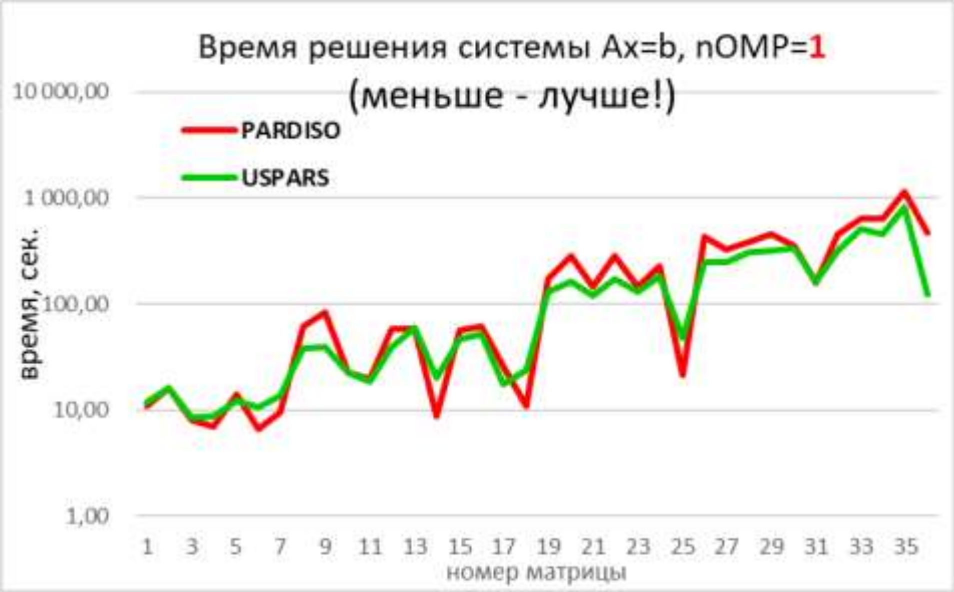
#	имя матрицы	размер	число ненулевых	число обусловленности
1	apache2	715 176	2 766 523	3,05E+06
2	torso3	259 156	4 429 042	2,27E+02
3	c-big	345 241	1 343 126	4,05E+04
4	ldoor	952 203	23 737 339	1,69E+08
5	F1	343 791	13 590 452	3,52E+06
6	thermal2	1 228 045	4 904 179	4,30E+06
7	G3_circuit	1 585 478	4 623 152	1,39E+07
8	H2O	67 024	1 141 880	4,24E+03
9	sparsine	50 000	799 494	2,78E+06
10	af_shell10	1 508 065	27 090 195	1,41E+10
11	boneS10	914 898	28 191 660	1,03E+08
12	appu	14 000	1 853 104	1,71E+02
13	PFlow_742	742 793	18 940 627	1,02E+12
14	memchip	2 707 524	14 810 202	1,41E+07
15	dielFilterV3real	1 102 824	45 204 422	8,67E+06
16	dielFilterV2real	1 157 456	24 848 204	8,91E+06
17	TSOPF_RS_b2383_c1	38 120	16 171 169	2,20E+08
18	Freescale1	3 428 755	18 920 347	1,03E+10

#	имя матрицы	размер	число ненулевых	число обусловленности
19	StocF-1465	1 465 137	11 235 263	2,41E+13
20	Ga10As10H30	113 081	3 114 357	2,90E+05
21	bone010	986 703	36 326 514	4,42E+08
22	Ge99H100	112 985	4 282 190	6,40E+03
23	Flan_1565	1 564 794	59 485 419	1,21E+08
24	audikw_1	943 695	39 297 771	1,83E+10
25	rajat31	4 690 002	20 316 253	3,33E+06
26	Ga19As19H42	133 123	4 508 981	3,07E+06
27	Hook_1498	1 498 023	31 207 734	3,60E+06
28	atmosmodl	1 489 752	10 319 760	1,05E+03
29	atmosmodd	1 270 432	8 814 880	4,88E+03
30	Transport	1 602 111	23 500 731	1,23E+06
31	ML_Geer	1 504 002	110 879 972	6,32E+08
32	atmosmodj	1 270 432	8 814 880	6,04E+03
33	Geo_1438	1 437 960	32 297 325	1,14E+13
34	vas_stokes_1M	1 090 664	34 767 207	1,73E+07
35	Serena	1 391 349	32 961 525	1,11E+14
36	circuit5M	5 558 326	59 524 291	3,43E+10

X86.USPARS vs. PARDISO: точность решения

Качество решения и используемая память USPARS соизмерима с конкурентами

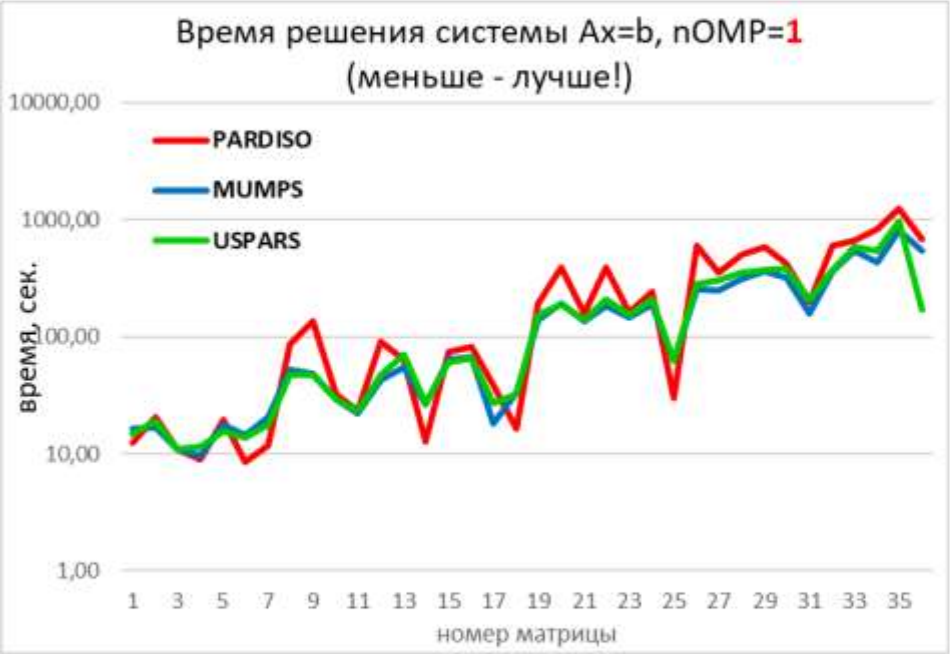
USPARS vs. PARDISO: производительность (AMD, 128 ядер)

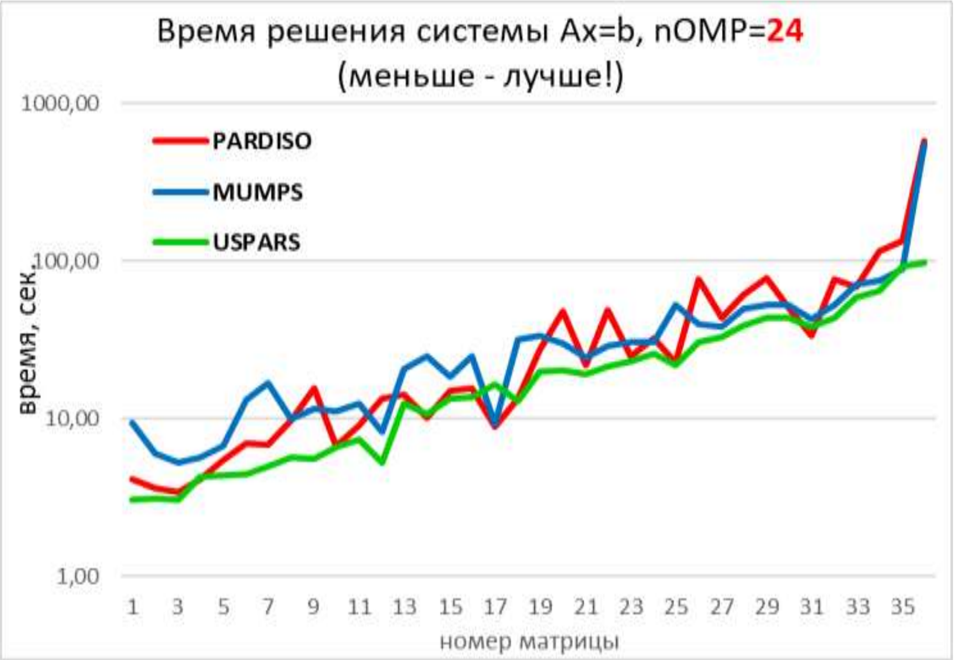


Среднее геометрическое отношений времен (Geomean)
больше единицы-лучше!
1,14

Среднее геометрическое отношений времен (Geomean)
больше единицы-лучше!
1,28

USPARS vs. PARDISO: производительность (Intel, 24 ядра)





Среднее геометрическое отношений времен (Geomean)
больше единицы-лучше!
1,19 (vs PARDISO)
0,97 (vs MUMPS)

Среднее геометрическое отношений времен (Geomean)
больше единицы-лучше!
1,40 (vs PARDISO)
1,60 (vs MUMPS)

X86.USPARS/PARDISO/MUMPS: производительность

(AMD, 128 ядер)

	Geomean
nOMP	PARDISO
1	1.14
2	1.18
4	1.17
8	1.35
16	1.30
24	1.36
64	1.35
128	1.28

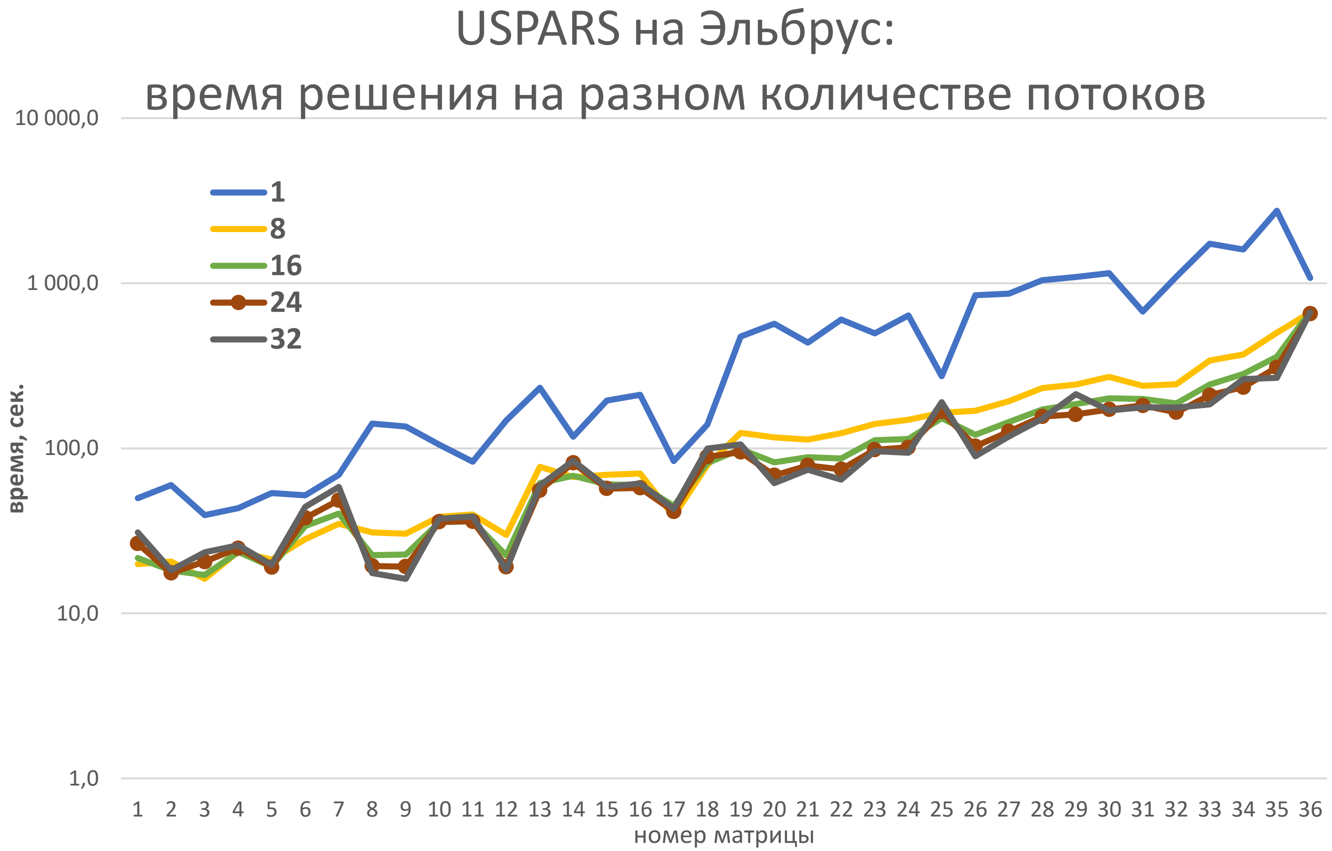
(Intel, 24 ядра)

	Geomean	
nOMP	PARDISO	MUMPS
1	1.19	0.97
8	1.37	1.37
24	1.40	1.60

Среднее геометрическое отношений времен (Geomean) = t_PARDISO/t_USPARS

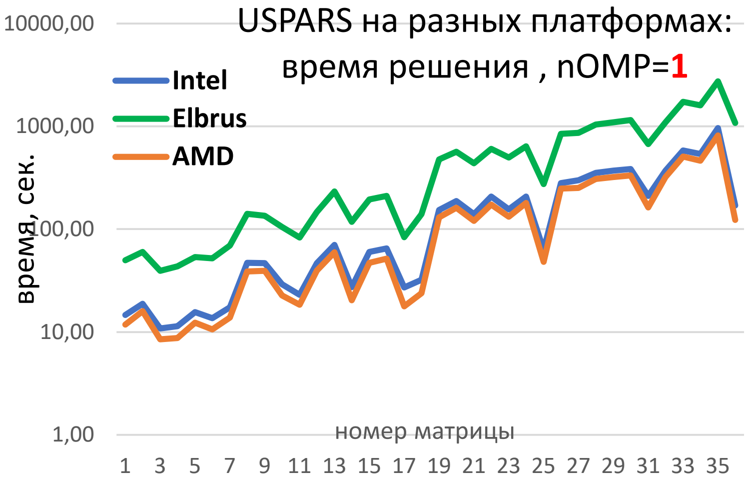
Чем больше результат, тем лучше!

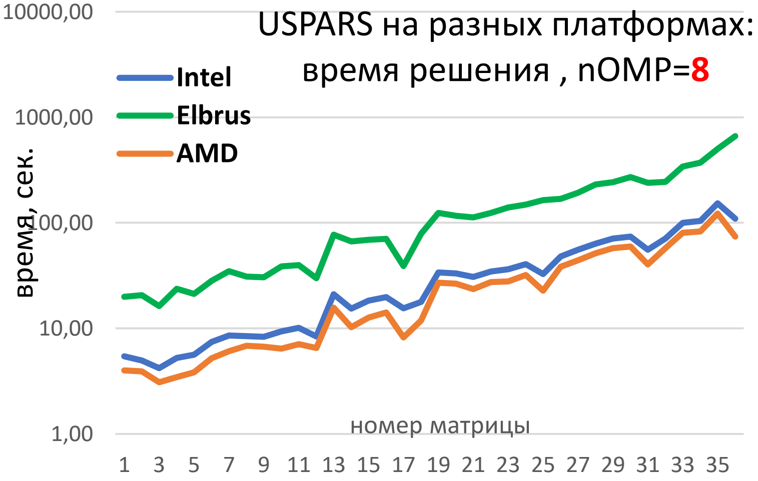
USPARS: Производительность на Elbrus

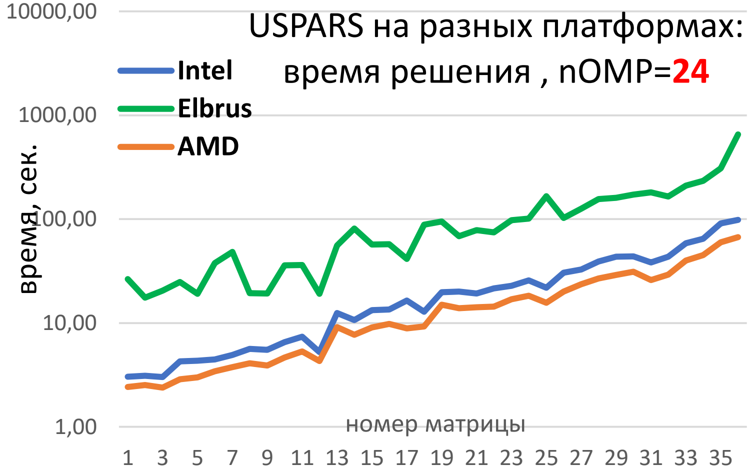


nOMP	Time ratio: 1OMP/nOMP Min	Max
1	1,00	1,00
2	1,11	1,92
4	1,40	3,48
8	1,62	5,47
12	1,65	6,96
16	1,53	7,65
20	1,47	9,32
24	1,37	8,87
32	1,17	10,26

USPARS: X86 & Elbrus







Среднее геометрическое отношений времен (Geomean)

	USPARS: Elbrus/	
nOMP	Intel	AMD
1	3,34	4,10
8	3,79	5,10
24	4,65	6,54

Спасибо за внимание!

Тел: +7 (383) 373 2463
Почта: marketing@unipro.ru

Новосибирский Центр Информационных Технологий «УНИПРО»