Izveštaj o eksperimentima jakog i slabog skaliranja

Hardverska i softverska arhitektura

- Procesor: Intel Core i7-1165G7 (11th Gen), 4 jezgra, 8 logičkih niti, bazni takt 2.8 GHz
- Cache memorija: 12 MB Intel® Smart Cache
- Broj jezgara : 4 jezgra/8 logičkih
- RAM memorija: 32.0 GB (31.7 GB usable)
- Operativni sistem: Widnows 10 Pro 64-bit
- Python 3.13.3
  Biblioteke:
  - numpy 2.3.2
  - multiprocessing
  - scikit-learn 1.7.1
- Rust 1.83.0
  Biblioteke
  - rayon 1.8
  - plotters 1.8
  - rand 0.9.2

Procenat paralelizacije

Značajan deo algoritma se može paralelizovati. Delovi koji se mogu paralelizovati:

- **Dodela tačaka klasterima** - Svaka tačka se nezavisno poredi sa svim centroidima → potpuno paralelizovano (najveći deo vremena).
- **Izračunavanje suma i broja tačaka po klasteru** - Može se raditi lokalno po niti (partial reduction), pa zatim kombinovati.
- **Računanje euklidskih rastojanja** - Svaka distanca se računa nezavisno, što znači da se kompletna matrica udaljenosti može računati u paraleli.

Sekvencijalni deo algoritma:

- Inicijalizacija centroida - Radi se jednom na početku, nije paralelizovano.
- Globalno ažuriranje centroida nakon svake iteracije
- Provera uslova konvergencije

Procena: oko 90% koda se može paralelizovati (dodela klastera + računanje suma + rastojanja).

Teorijski maksimum ubrzanja (Amdalov i Gustavsonov zakon)

Amdalov zakon (jako skaliranje):

- $\text{speedup} = 1 / (s + p / N)$
- primer → p = 0.9, s=0.1, N = 4 → speedup = 1/0.1 + 0.225 = 1/0.325 = 3.07

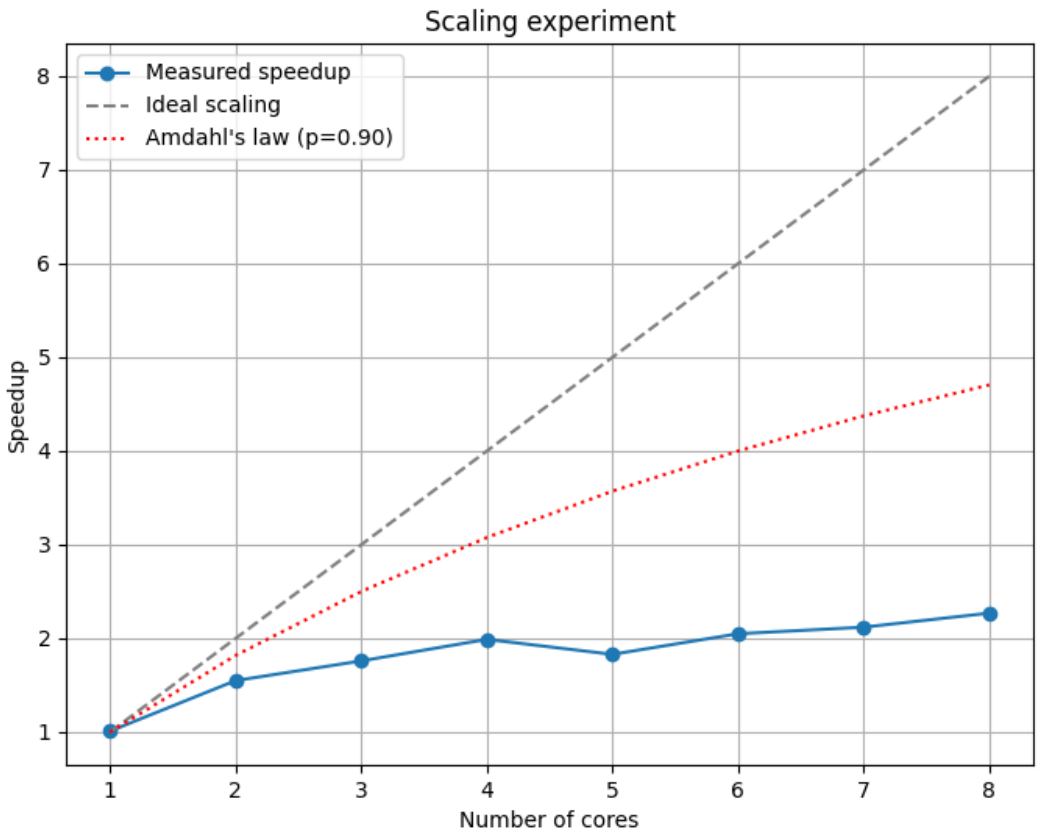
Gustafsonov zakon (slabo skaliranje):

- scaled speedup = $s + p \times N$
- primer → p = 0.9, s=0.1, N=4 → scaled speedup = 0.1 + 0.9*4 = 3.7

Jako skaliranje Python

Broj tačaka koje se klasterizuju je kontantan (npr. 100 000), K = 8.

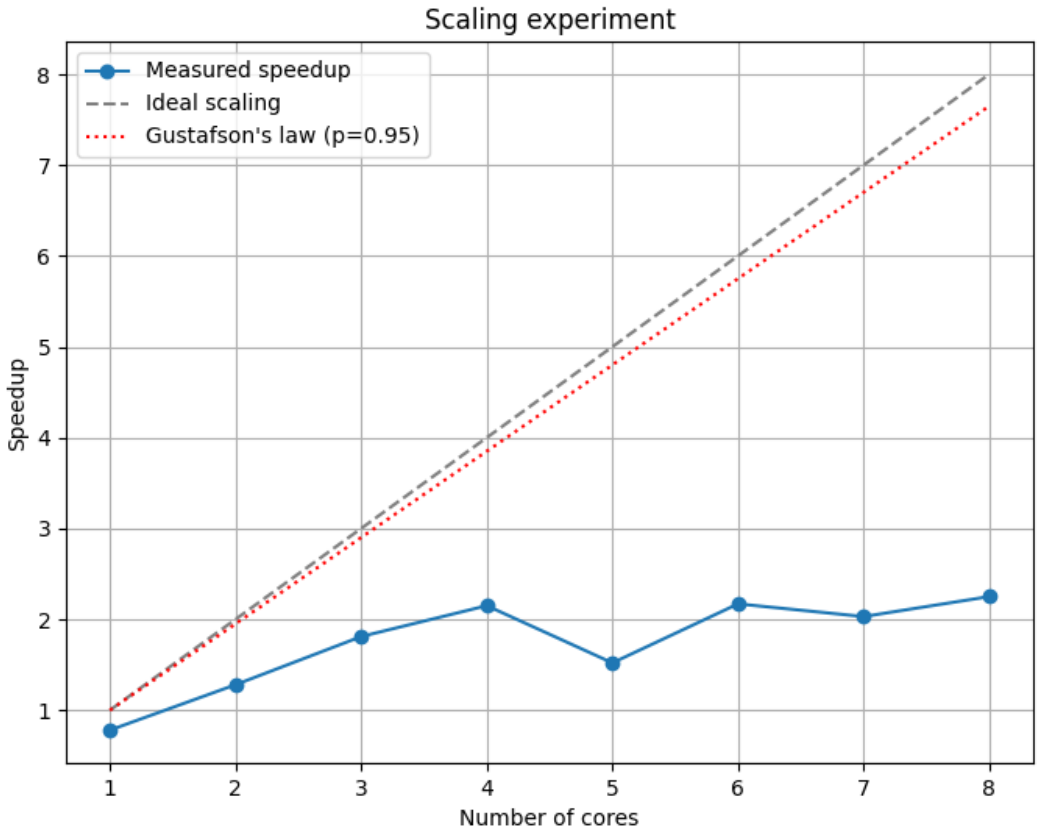
Threads	Samples num	Mean Seq	Std Seq	Mean Par	Std Par	Speedup
1	100 000	166.7599	0.5092	164.3769	2.5429	1.01
2	100 000	180.4955	0.7007	116.4167	1.1565	1.55
3	100 000	184.4546	0.437	104.6522	0.643	1.76
4	100 000	185.2969	0.3809	93.062	0.707	1.99
5	100 000	192.3256	12.8893	105.1112	0.5529	1.83
6	100 000	205.2563	2.0234	100.0328	0.4419	2.05
7	100 000	205.0856	1.202	96.9064	0.2316	2.12
8	100 000	204.4711	0.6962	90.1254	0.2111	2.27



Slabo skaliranje Python

Posao po jezgru se ne menja. Sa povećanjem broja jezgara povećava se i broj tačaka za klasterizaciju. K=8.

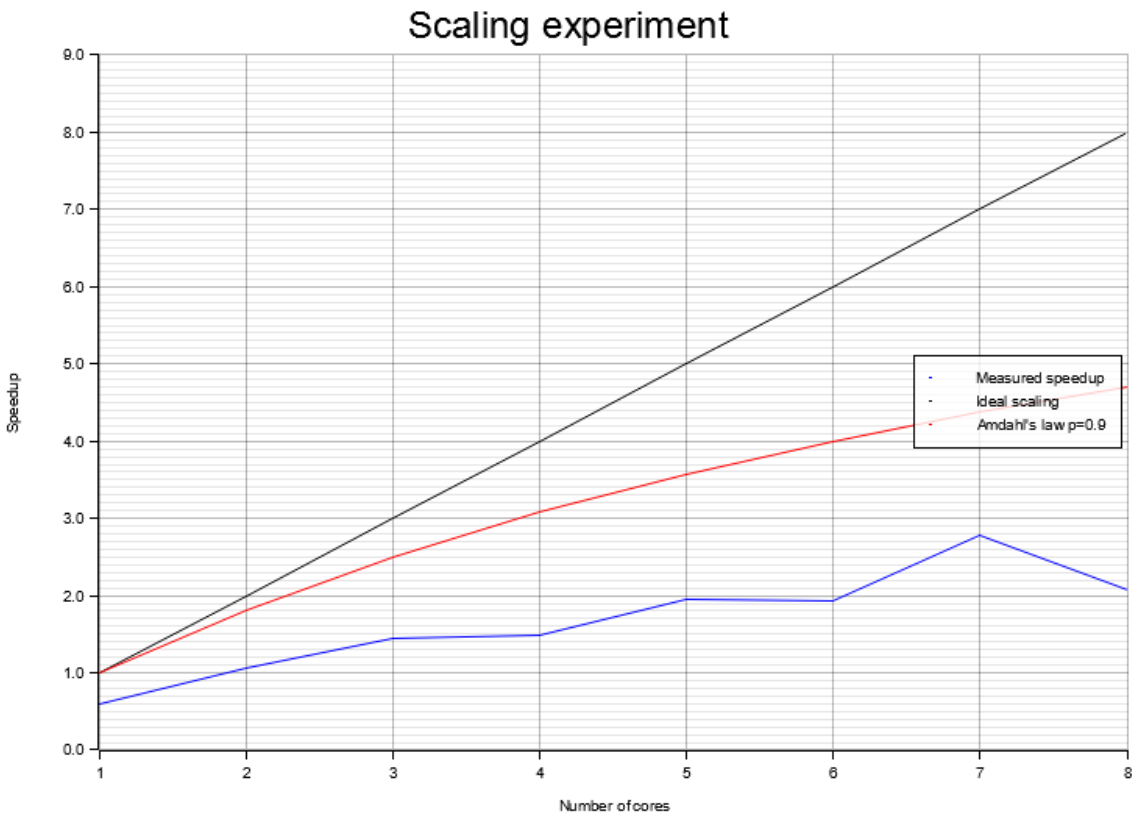
Threads	Samples num	Mean Seq	Std Seq	Mean Par	Std Par	Speedup
1	50 000	29.9706	0.3032	38.295	0.2097	0.78
2	100 000	85.7446	0.0502	66.8344	0.3158	1.28
3	150 000	223.6307	0.1651	123.3185	0.7614	1.81
4	200 000	398.4888	0.4253	185.6973	0.4383	2.15
5	250 000	102.0701	0.0858	67.255	0.3864	1.52
6	300 000	601.8269	0.8173	276.956	0.5364	2.17
7	350 000	657.8125	0.4331	324.1145	0.1647	2.03
8	400 000	765.8667	0.1082	339.7227	0.2273	2.25



Jako skaliranje Rust

Broj tačaka koje se klasterizuju je kontantan (npr. 1 000 000), K=4.

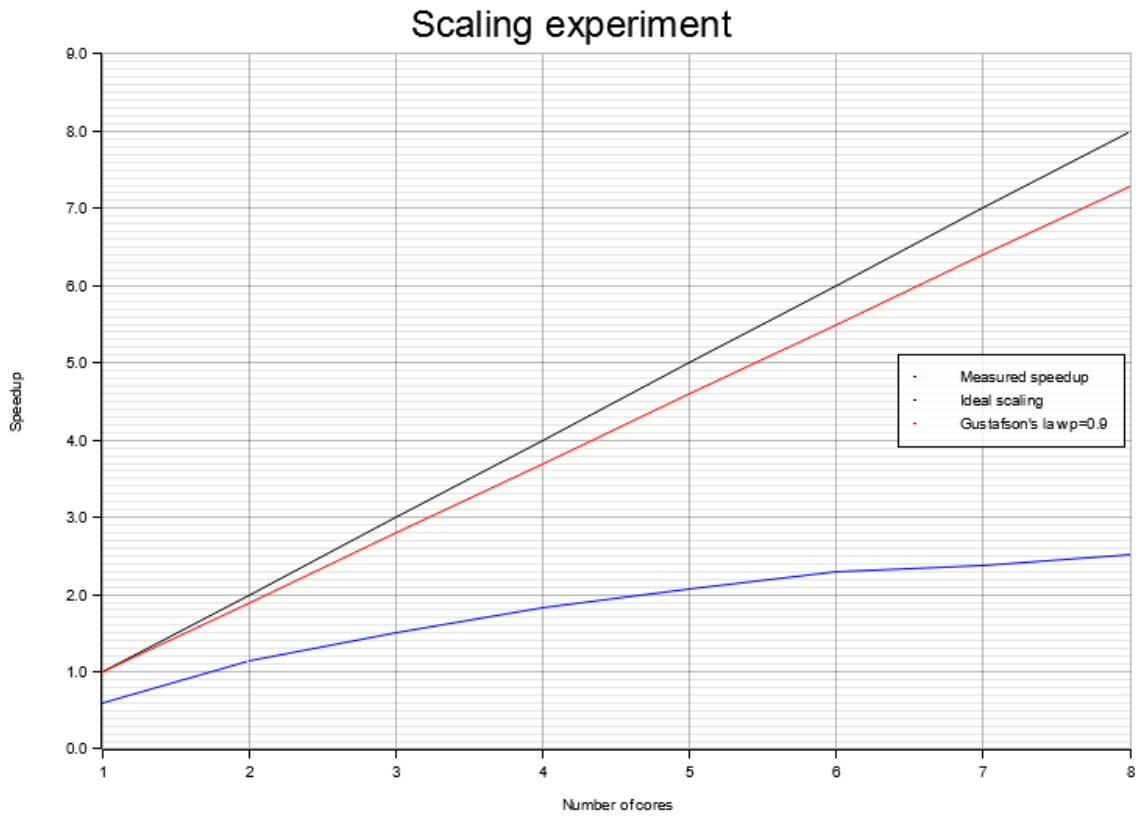
Threads	Samples num	Mean Seq	Std Seq	Mean Par	Std Par	Speedup
1	1 000 000	4.8061	1.018	7.9874	1.5684	0.6
2	1 000 000	4.4009	0.2979	4.1148	0.3272	1.07
3	1 000 000	4.4941	0.4664	3.1178	0.2396	1.44
4	1 000 000	5.9519	1.6545	4.0184	0.9137	1.48
5	1 000 000	6.2461	0.6669	3.1973	0.9461	1.95
6	1 000 000	5.2452	0.1925	2.7173	0.192	1.93
7	1 000 000	5.3825	0.0229	1.934	0.0745	2.78
8	1 000 000	5.7609	0.1893	2.7869	0.0394	2.07



Slabo skaliranje Rust

Posao po jezgru se ne menja. Sa povećanjem broja jezgara povećava se i broj tačaka za klasterizaciju. K=4.

Threads	Samples num	Mean Seq	Std Seq	Mean Par	Std Par	Speedup
1	200 000	1.7926	0.1235	2.9876	0.2386	0.6
2	400 000	3.6449	0.3783	3.225	0.346	1.13
3	600 000	4.9185	1.0333	3.2358	1.0974	1.52
4	800 000	7.5401	0.4145	4.1202	0.2629	1.83
5	1 000 000	10.4781	0.4418	5.0375	0.505	2.08
6	1 200 000	15.6332	0.9253	6.797	0.1272	2.3
7	1 400 000	11.3857	0.3156	4.7638	0.2637	2.39
8	1 600 000	16.0841	2.2382	6.3573	1.173	2.53



Kod eksperimenta slabog skaliranja manipuliše se veličinom ulaznog skupa podataka tako da se sa povećanjem broja procesorskih jezgara proporcionalno povećava i broj podataka koji se obrađuju. Na taj način, posao po jezgru ostaje konstantan, odnosno svako jezgro dobija približno isti broj tačaka za obradu bez obzira na ukupan broj jezgara. U implementaciji se to postiže tako što se broj tačaka (n_points) određuje kao proizvod broja jezgara (threads) i baznog broja tačaka (base_points). Na primer, za 1 jezgro se obrađuje 200.000 tačaka, za 2 jezgra 400.000 itd. Time se obezbeđuje da se sa svakim dodatnim jezgrom ukupna količina posla povećava. Ovakav pristup omogućava da se meri efikasnost

paralelizacije kada se resursi skaliraju, a posao po jezgru ostaje konstantan, što je osnovna ideja slabog skaliranja.