Високопроизводителни методи и алгоритми за оптимизационни задачи

Иван Лирков

1 декември 2021 г.

Оптимизационна задача

Нека $\mathbf{f} \in [0, \nu]^{M \times N}$ е двумерно изображение с размери $M \times N$ и $n = MN$. Решаваме следната оптимизационна задача

$$\underset{\mathbf{u} \in [0,\nu]^n}{\operatorname{argmin}} \|\nabla \mathbf{u}\|_{2,1} \quad \text{subject to} \quad \|T(\mathbf{u}) - T(\mathbf{f})\|_2^2 \leq n,$$

$\nabla \in \mathbb{R}^{2n \times n}$ – оператор (дискретен градиент)
$\|\cdot\|_{2,1}$ – TV (total variation) полу-норма (сума от дължините на градиентите)
T – трансформация на Анскомб:

$$\begin{aligned} T &: [0, +\infty)^n \to (0, +\infty)^n \\ &: \mathbf{v} = (v_i)_{1 \leq i \leq n} \mapsto \left(2\sqrt{v_i + \frac{3}{8}} \right)_{1 \leq i \leq n}. \end{aligned}$$

Алгоритъм

Първият алгоритъм е описан в

S. Harizanov, J.-C. Pesquet, G. Steidl. Epigraphical projection for solving least squares Anscombe transformed constrained optimization problems. *Scale-Space and Variational Methods in Computer Vision*, *Lecture Notes in Computer Science*, volume 7893, 125–136. Springer, 2013.
и има следния вид:

Алгоритъм

$$\mathbf{u}^{(0)}, \boldsymbol{\zeta}^{(0)}, \left(\mathbf{p}_j^{(0)}\right)_{1\le j\le 3} = \left(\bar{\mathbf{p}}_j^{(0)}\right)_{1\le j\le 3}.$$

For $k = 0, 1, \ldots$ repeat until a stopping criterion is reached

1. $\mathbf{u}^{(k+1)} = \max\left\{\min\left\{\left(\mathbf{u}^{(k)} - \sigma\rho\left(\bar{\mathbf{p}}_1^{(k)} + \nabla^*\bar{\mathbf{p}}_2^{(k)}\right)\right), \nu\mathbf{1}_n\right\}, \mathbf{0}\right\}$
2. $\boldsymbol{\zeta}^{(k+1)} = P_{V_n}\left(\boldsymbol{\zeta}^{(k)} - \sigma\rho\bar{\mathbf{p}}_3^{(k)}\right)$
3. $(v_{1,i}, \eta_i) = P_{\operatorname{epi}\varphi_i}\left(p_{1,i}^{(k)} + \left(\mathbf{u}^{(k+1)}\right)_i + 3/8\,,\; p_{3,i}^{(k)} + \zeta_i^{(k+1)}\right),$
4. $\mathbf{v}_2 = \mathbf{p}_2^{(k)} + \nabla\mathbf{u}^{(k+1)}$
5. $\mathbf{p}_1^{(k+1)} = \mathbf{p}_1^{(k)} + \mathbf{u}^{(k+1)} + 3/8 - \mathbf{v}_1$
6. $\mathbf{p}_2^{(k+1)} = \mathbf{v}_2 - \operatorname{prox}_{\sigma^{-1}\|\cdot\|_{2,1}}(\mathbf{v}_2)$
7. $\mathbf{p}_3^{(k+1)} = \mathbf{p}_3^{(k)} + \boldsymbol{\zeta}^{(k+1)} - \boldsymbol{\eta}$
8. $\bar{\mathbf{p}}_j^{(k+1)} = \mathbf{p}_j^{(k+1)} + \left(\mathbf{p}_j^{(k+1)} - \mathbf{p}_j^{(k)}\right), \quad j = 1, 2, 3.$

Алгоритъм

В стъпка 1 — проекция върху хиперкуба $[0, \nu]^n \subset \mathbb{R}^n$

В стъпка 2 — проекция върху затвореното полупространство $V_n := \{\boldsymbol{\zeta} \in \mathbb{R}^n : \langle \mathbf{1}_n, \boldsymbol{\zeta} \rangle \leq n\}$

В стъпка 3 — епиграфска проекция върху

$$\varphi_i(x) = \begin{cases} (2\sqrt{x} - f_i)^2 & \text{if } x \geq 0, \\ +\infty & \text{otherwise,} \end{cases}$$

В стъпка 6 — свиване с праг σ^{-1}.

Алгоритъм

Вторият алгоритъм е описан в

S. Harizanov, Reconstructing 2D radiographic data. tech. rep., University of Szeged; Szeged: 2017.

и има следния вид:

Алгоритъм

I. Preprocessing: $\mathbf{f}_A = T(\mathbf{f})$ and $\tau_A = n$.

II. Initialization: $\mathbf{u}_A^{(0)} = 0$, $\bar{\mathbf{p}}_1^{(0)} = \mathbf{p}_1^{(0)} = 0$, $\bar{\mathbf{p}}_2^{(0)} = \mathbf{p}_2^{(0)} = 0$.

III. For $k = 0, 1, \ldots, K$

1. $\mathbf{u}_A^{(k+1)} = P_{[T(0),T(\omega)]}\left(\underbrace{\mathbf{u}_A^{(k)} - \sigma\rho\left(\bar{\mathbf{p}}_1^{(k)} + \nabla^* \bar{\mathbf{p}}_2^{(k)}\right)}_{x}\right)$
2. $v_1^{(k+1)} = P_{B_N(\mathbf{f}_A, \sqrt{\tau_A})}\left(\mathbf{u}_A^{(k+1)} + \mathbf{p}_1^{(k)}\right) =$
$$\begin{cases} \mathbf{u}_A^{(k+1)} + \mathbf{p}_1^{(k)}, & \text{if } \left\|\mathbf{u}_A^{(k+1)} + \mathbf{p}_1^{(k)} - \mathbf{f}_A\right\|_2 \leq \sqrt{\tau_A} \\ \frac{\sqrt{\tau_A}\left(\mathbf{u}_A^{(k+1)} + \mathbf{p}_1^{(k)} - \mathbf{f}_A\right)}{\left\|\mathbf{u}_A^{(k+1)} + \mathbf{p}_1^{(k)} - \mathbf{f}_A\right\|_2} + \mathbf{f}_A, & \text{otherwise} \end{cases}$$
3. $\mathbf{v}_2 = \mathbf{p}_2^{(k)} + \nabla \mathbf{u}_A^{(k+1)}$
4. $\mathbf{p}_1^{(k+1)} = \mathbf{p}_1^{(k)} + \mathbf{u}_A^{(k+1)} - \mathbf{v}_1^{(k+1)}$
5. $\mathbf{p}_2^{(k+1)} = \mathbf{v}_2 - \operatorname{prox}_{\sigma^{-1}\|\cdot\|_{2,1}}(\mathbf{v}_2)$

Алгоритъм

I. Preprocessing: $\mathbf{f}_A = T(\mathbf{f})$ and $\tau_A = n$.

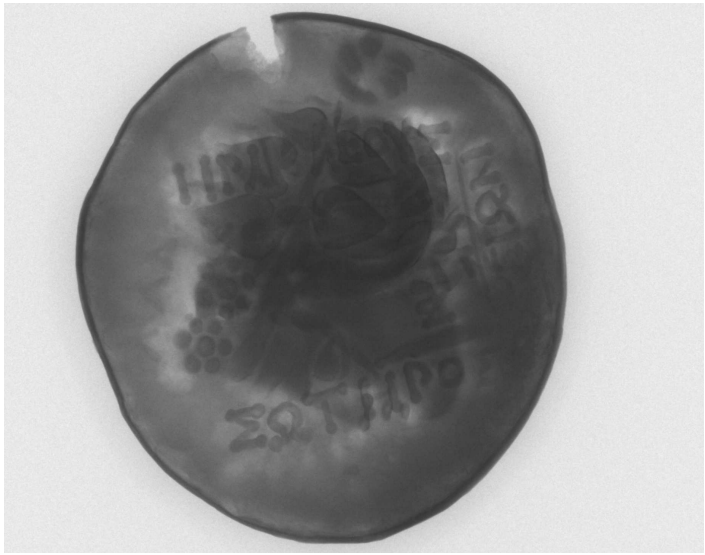
II. Initialization: $\mathbf{u}_A^{(0)} = 0$, $\bar{\mathbf{p}}_1^{(0)} = \mathbf{p}_1^{(0)} = 0$, $\bar{\mathbf{p}}_2^{(0)} = \mathbf{p}_2^{(0)} = 0$.

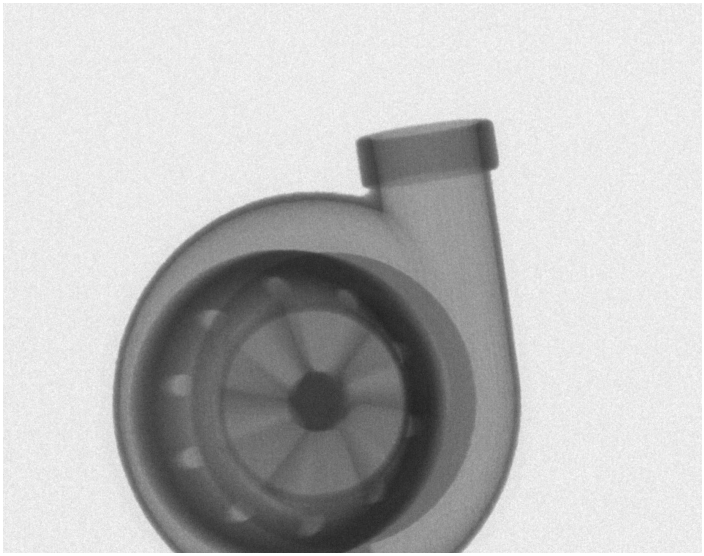
III. For $k = 0, 1, \ldots, K$

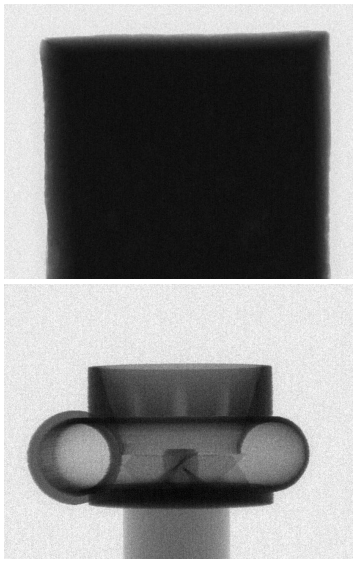
1. $\mathbf{u}_A^{(k+1)} = P_{[T(0),T(\omega)]}\left(\underbrace{\mathbf{u}_A^{(k)} - \sigma\rho\left(\bar{\mathbf{p}}_1^{(k)} + \nabla^* \bar{\mathbf{p}}_2^{(k)}\right)}_{x}\right)$
2. $v_1^{(k+1)} = P_{B_N(\mathbf{f}_A, \sqrt{\tau_A})}\left(\mathbf{u}_A^{(k+1)} + \mathbf{p}_1^{(k)}\right)$
3. $\mathbf{v}_2 = \mathbf{p}_2^{(k)} + \nabla \mathbf{u}_A^{(k+1)}$
4. $\mathbf{p}_1^{(k+1)} = \mathbf{p}_1^{(k)} + \mathbf{u}_A^{(k+1)} - \mathbf{v}_1^{(k+1)}$
5. $\mathbf{p}_2^{(k+1)} = \mathbf{v}_2 - \mathrm{prox}_{\sigma^{-1}\|\cdot\|_{2,1}}(\mathbf{v}_2)$
6. $\bar{\mathbf{p}}_i^{(k+1)} = \mathbf{p}_i^{(k+1)} + \left(\mathbf{p}_i^{(k+1)} - \mathbf{p}_i^{(k)}\right)$, $i = 1, 2$.

IV. Post-processing: $\bar{\mathbf{u}} = T^{-1}\left(\mathbf{u}_A^{(K)}\right)$.

Входни изображения







1840×1446

920×723

Паралелна реализация

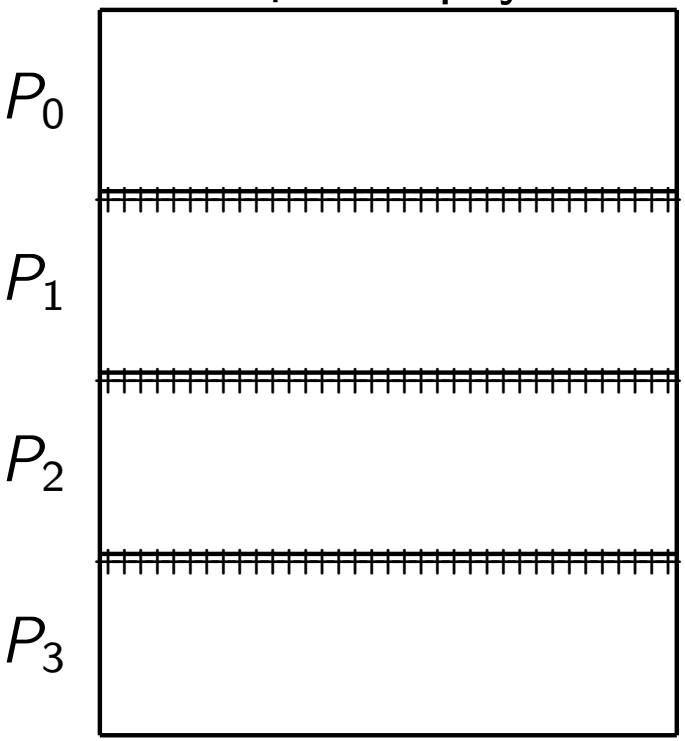
- Многонишкова паралелна реализация
  От изчислителна гледна точка алгоритъм 1 се състои от метод на Нютон в стъпка 3, скаларно умножение и векторни суми във всички стъпки.
  Използваме OpenMP за компонентните епиграфски проекции (стъпка 3) и свиване (стъпка 6), както и за всички векторни операции.

Паралелна реализация

- Многонишкова паралелна реализация
- Реализация върху възли с разпределена памет
  Разделяме изображението на m правоъгълници, така че всеки правоъгълник да съдържа приблизително MN/m пиксела. Един изчислителен възел изчислява всички пиксели от даден правоъгълник. По този начин методът на Нютон се изпълнява паралелно за всеки пиксел и не се нуждае от никаква комуникация. Същото важи и за скаларното умножение и векторните суми.

Паралелна реализация

- Многонишкова паралелна реализация
- Реализация върху възли с разпределена памет



Разпределение на данни на четири процесора.
Изчисляването на оператора за дискретен градиент изисква само M стойности от следващия възел.

Паралелна реализация

- Многонишкова паралелна реализация
- Реализация върху възли с разпределена памет
  Изчисляването на оператора за дискретен градиент изисква само M стойности от следващия възел. За подобряване на производителността, използвам припокриване на изчисления и локална комуникация. Само изчисляването на проекцията върху V_n (стъпка 2) изисква глобална комуникация. Функцията MPI_Allreduce е използвана за изчисляване на скаларното произведение $\langle \mathbf{1}_n, \boldsymbol{\zeta} \rangle$.

Компютърна система

- Авитохол
  HP Cluster Platform SL250S GEN8, 150 servers
  two 8-core Intel Xeon E5-2650 v2 8C processors at 2.6 GHz and two Intel Xeon Phi 7120P coprocessors
  64 GB of memory per node
  16 GB of memory per coprocessor
  high-speed InfiniBand FDR network
  Intel MPI Library 2017 Update 2
  Intel C Compiler 17.0.2

Време за изпълнение

Времето за изпълнение в секунди на един възел на Авитохол.
(Средно време за изпълнение за 100 000 итерации)

M	N	number of threads on one node					
		1	2	4	8	16	32
723	920	5520.07	4371.80	2294.13	1288.48	849.77	557.19
1446	1840	21547.00	18157.96	9400.75	5299.85	3576.14	3052.74
1840	1446	21057.86	18033.88	9389.83	5310.77	3581.53	2528.93

Време за изпълнение

Времето за изпълнение в секунди на много възли на Авитохол.

M	N	number of nodes					
		2	3	4	5	6	8
16 threads							
723	920	384.70	233.90	**151.42**	**117.97**	**99.86**	**77.60**
1446	1840	1764.39	1162.12	864.54	699.46	550.16	397.40
1840	1446	1785.73	1160.34	862.35	**688.89**	**546.60**	**393.59**
32 threads							
723	920	**246.15**	**207.84**	184.13	174.62	192.41	191.02
1446	1840	**1248.47**	**1137.24**	**815.63**	**671.48**	**523.21**	**330.31**
1840	1446	**1619.59**	**1139.02**	**851.25**	733.85	594.25	431.20

Време за изпълнение

Време за изпълнение в секунди на Xeon Phi (Авитохол).

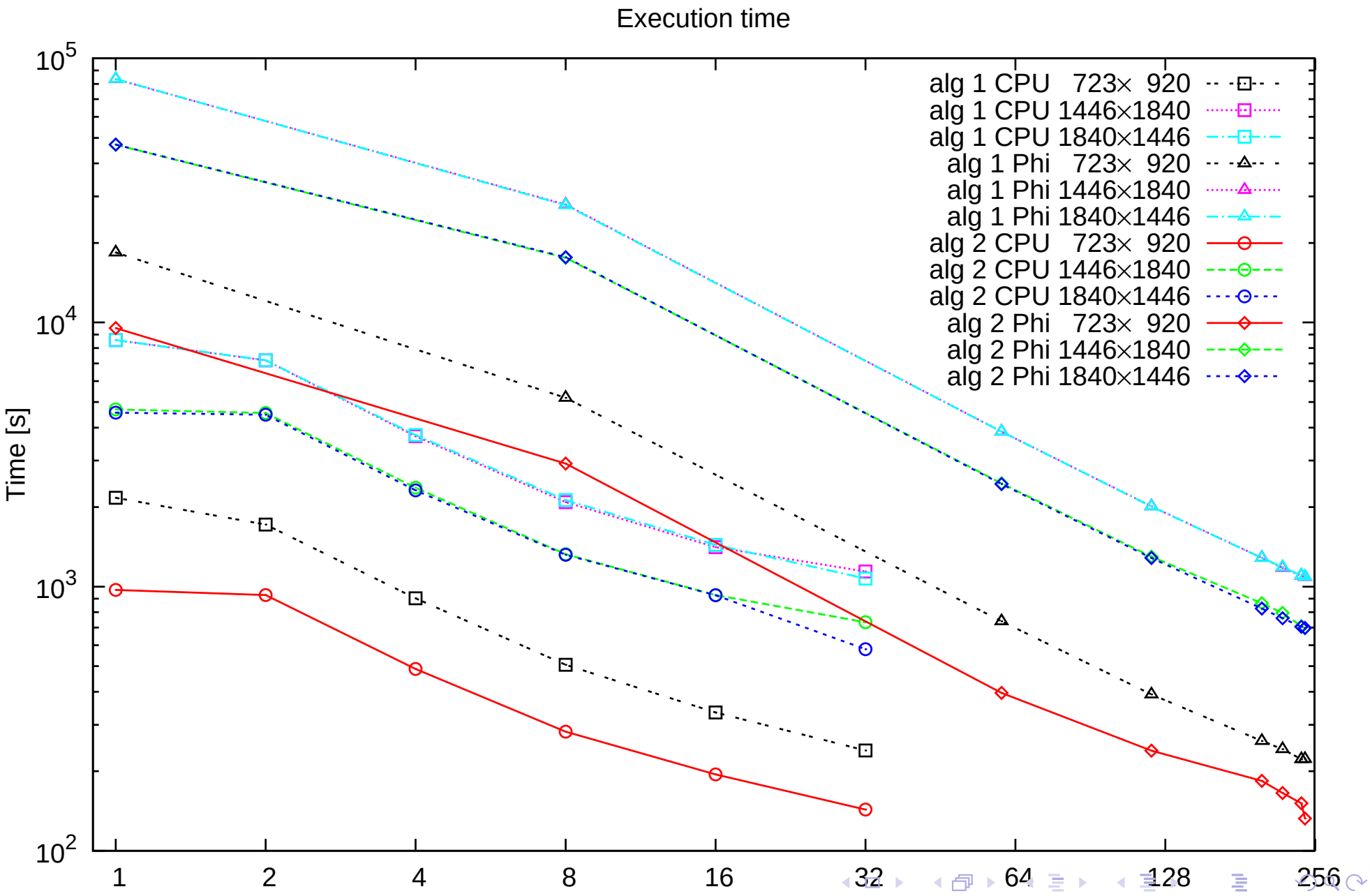
M	N	number of threads on one Xeon Phi						
		1	8	60	120	240	244	
723	920	45176.10	8863.91	1340.72	822.56	550.51	549.88	
1446	1840	205412.67	37072.82	6219.42	3818.24	2614.77	2599.64	
1840	1446	208701.67	69456.75	9654.39	5043.79	2754.97	2719.17	
		number of nodes (2 processes per node, 244 threads)						
		1	2	3	4	5	6	8
723	920	**321.86**	**207.80**	**168.87**	153.55	**145.73**	**137.03**	128.13
1446	1840	**1336.17**	**674.21**	**488.44**	**402.27**	**352.23**	**317.99**	**277.76**
1840	1446	**1338.11**	**664.77**	**479.10**	**384.34**	**335.10**	**311.14**	**257.27**

Време за изпълнение

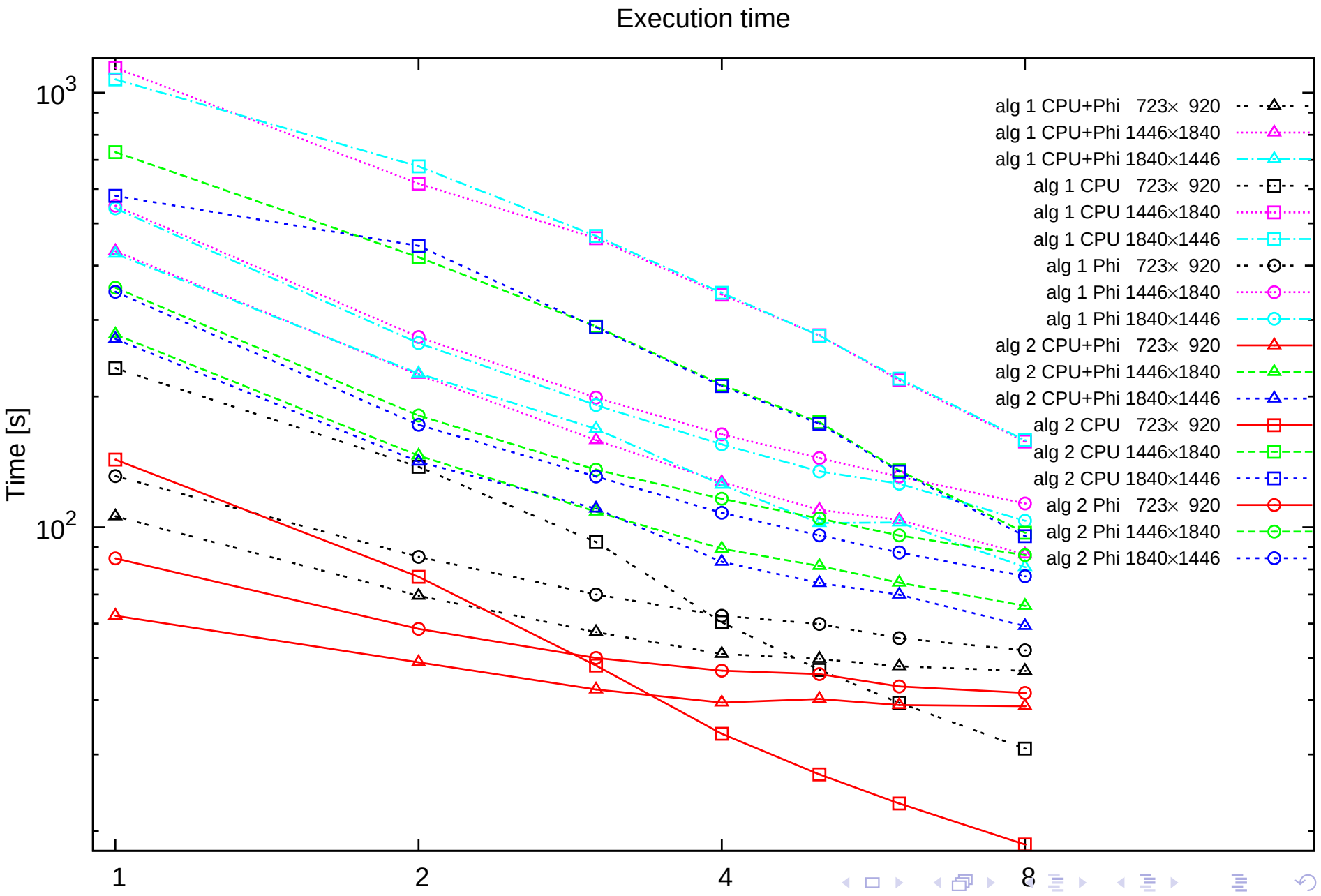
Време за изпълнение в секунди върху процесорите и копроцесорите на Авитохол.

M	N	nodes						
		1	2	3	4	5	6	8
723	920	260.04	167.25	137.54	123.14	118.85	121.29	110.98
1446	1840	1110.95	554.87	394.63	318.25	273.65	267.12	215.83
1840	1446	1091.79	561.22	409.55	310.96	254.32	264.09	201.53

Време за изпълнение



Време за изпълнение



Време за изпълнение