Performance Analysis of a Parallel Denoising Algorithm on Intel Xeon Computer System

Ivan Lirkov

Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, BULGARIA

http://www.iict.bas.bg/EN/

ivan@parallel.bas.bg

Outline

- Constrained optimization problem

Outline

- Constrained optimization problem
- Algorithm

Outline

- Constrained optimization problem
- Algorithm
- Input images

Outline

- Constrained optimization problem
- Algorithm
- Input images
- Parallel implementation

Outline

- Constrained optimization problem
- Algorithm
- Input images
- Parallel implementation
- Computer system

Outline

- Constrained optimization problem
- Algorithm
- Input images
- Parallel implementation
- Computer system
- Execution time and speed-up

Constrained optimization problem

Let us consider 2D image $\mathbf{f} \in [0,\nu]^{M\times N}$ and let us denote the size of the image by $n = MN$. In order to remove the noise from the image, we solve the following constrained optimization problem:

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

$\nabla \in \mathbb{R}^{2n\times n}$ is the discrete gradient operator
$\|\cdot\|_{2,1}$ denotes the total variation (TV) semi-norm (the sum of the lengths of the 2D gradient vectors)
T is the Anscombe transform:

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

Algorithm

The proposed algorithm is taken from

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.

and can be written as follows:

Algorithm

$\mathbf{u}^{(0)}, \boldsymbol{\zeta}^{(0)}, \left(\mathbf{p}_j^{(0)}\right)_{1\le j\le 3} = \left(\bar{\mathbf{p}}_j^{(0)}\right)_{1\le j\le 3}$, $(\rho,\sigma)\in(0,+\infty)^2$, $\rho\sigma < 1/9$.

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. $(v_{1,i}, \eta_i) = P_{\mathrm{epi}\,\varphi_i}\left(p_{1,i}^{(k)} + \left(\mathbf{u}^{(k+1)}\right)_i + 3/8\,,\, p_{3,i}^{(k)} + \zeta_i^{(k+1)}\right), \quad i = 1, \ldots, n$
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. $\mathbf{p}_2^{(k+1)} = \mathbf{v}_2 - \mathrm{prox}_{\sigma^{-1}\|\cdot\|_{2,1}}(\mathbf{v}_2)$
7. $\mathbf{p}_3^{(k+1)} = \mathbf{p}_3^{(k)} + \boldsymbol{\zeta}^{(k+1)} - \boldsymbol{\eta}$
8. $\bar{\mathbf{p}}_j^{(k+1)} = \mathbf{p}_j^{(k+1)} + \left(\mathbf{p}_j^{(k+1)} - \mathbf{p}_j^{(k)}\right), \quad j = 1, 2, 3.$

Algorithm

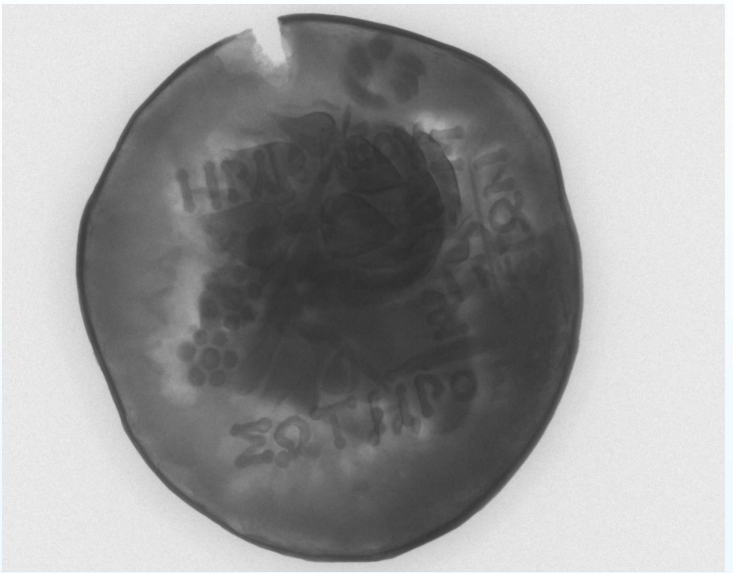
In Step 1 we perform projection onto the hypercube $[0, \nu]^n \subset \mathbb{R}^n$
In Step 2 — projection onto the closed half-space
$V_n := \{\boldsymbol{\zeta} \in \mathbb{R}^n : \langle \mathbf{1}_n, \boldsymbol{\zeta} \rangle \leq n\}$
In Step 3 — projection onto the epigraph of

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

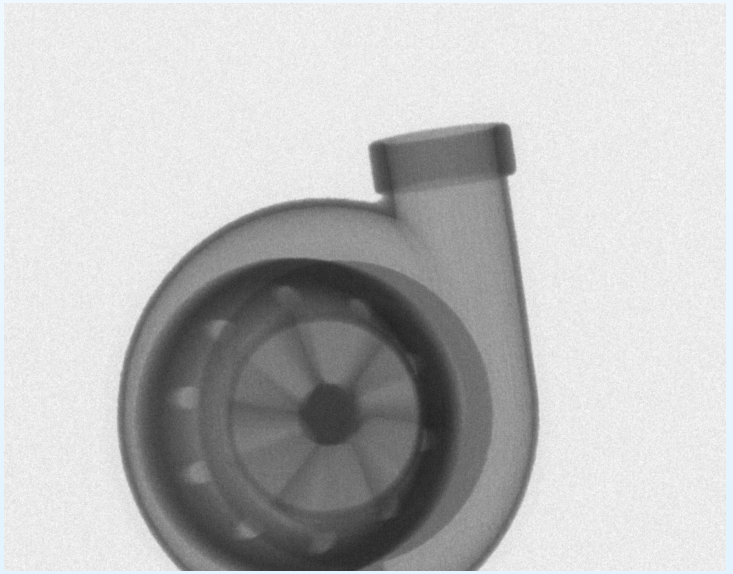
In Step 6 the coupled soft shrinkage with threshold σ^{-1} is performed.
The remaining steps can be computed in a straightforward way.

Input images

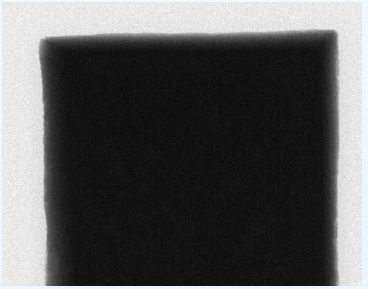
1840 × 1446



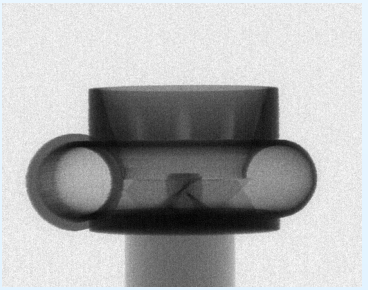
1840 × 1446



1840 × 1446



920 × 723



920 × 723

Parallel implementation

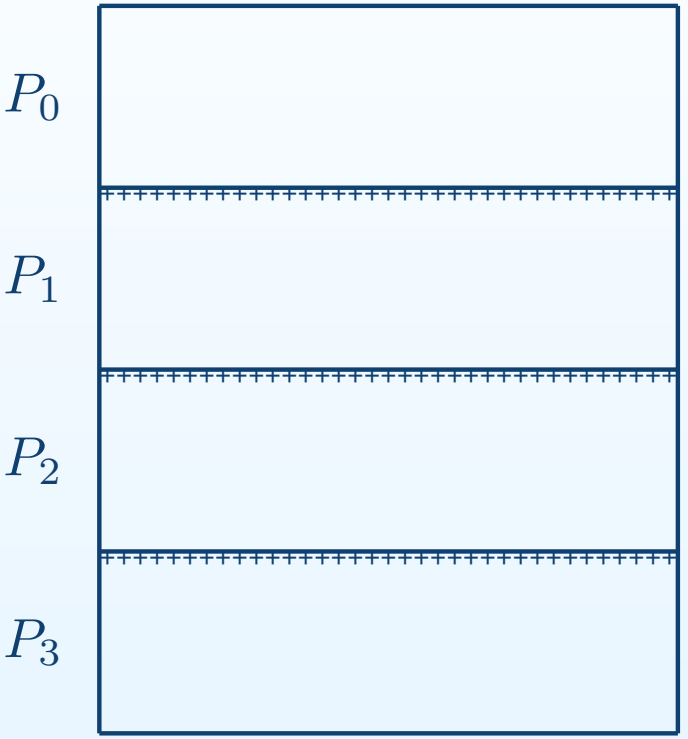
- Multi-thread-based parallel implementation
  From the computational point of view, the algorithm consists of Newton's method in step 3, scalar multiplication and vector sums in all steps.
  We use OpenMP for the component-wise epigraphical projections (step 3) and the coupled soft shrinkage (step 6) as well as for all vector operations.

Parallel implementation

- Multi-thread-based parallel implementation
- Multi-node implementation
  We partition the image into m rectangles so that each rectangle contains approximately MN/m pixels. We map all pixels from a given rectangle into a single computing node. In this way, the Newton's method is performed in parallel for each pixel and does not need any communication. The same is true for the scalar multiplication and vector sums.

Parallel implementation

- Multi-thread-based parallel implementation
- Multi-node implementation



Data distribution on four processors.
The computation of the discrete gradient operator requires only M values from the next node (denoted by $+$ in the figure).

Parallel implementation

- Multi-thread-based parallel implementation
- Multi-node implementation
  The computation of the discrete gradient operator requires only M values from the next node.
  To improve the performance, we have used overlapping of computation and local communication.
  Only the computation of the projection onto V_n (step 2) requires a single global communication. The function MPI_Allreduce was used to compute the inner product $\langle \mathbf{1}_n, \boldsymbol{\zeta} \rangle$.

Computer system

- Avitohol
  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

Execution time

The CPU time in seconds on one node of Avitohol.
(Average execution time for 100000 iterations)

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

Execution time

The CPU time in seconds on many nodes of Avitohol.

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

Execution time

The execution time in seconds on Xeon Phi (Avitohol).

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**

Execution time

The execution time in seconds on processors and coprocessors of the Avitohol.

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

Speed-up

Speed-up using only CPUs on Avitohol.

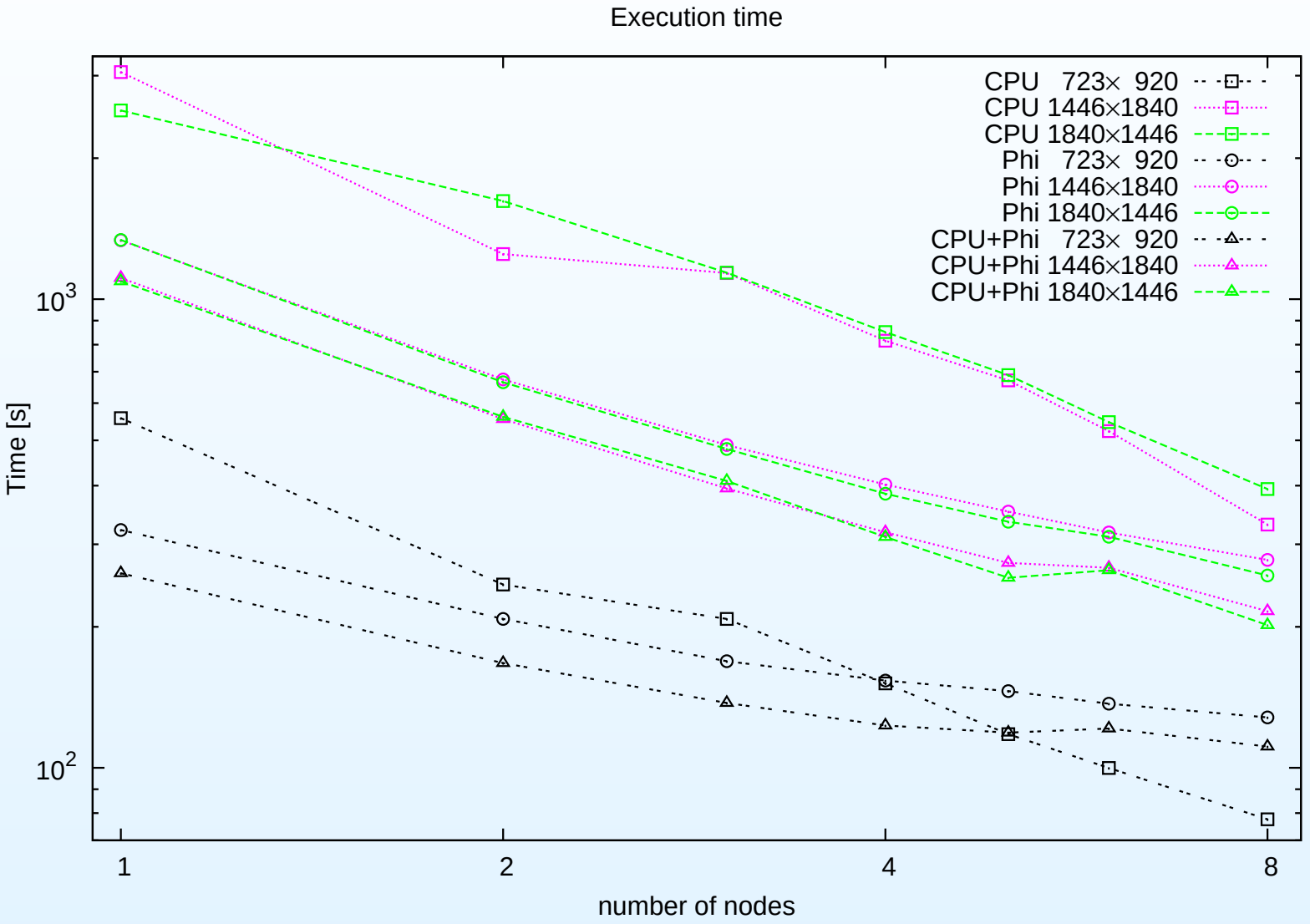
number of threads	$M \times N$		
	723×920	1446×1840	1840×1446
2	1.26	1.19	1.17
4	2.41	2.29	2.25
8	4.29	4.07	3.97
16	6.49	6.02	5.85
32	9.91	7.06	8.34
48	22.45	17.28	13.03
64	26.59	18.97	18.53
80	36.49	26.69	24.74
96	46.81	32.13	30.63
128	55.33	41.18	38.60
256	71.23	65.31	53.61

Speed-up

Speed-up using only coprocessors on Avitohol.

number of threads	$M \times N$		
	723×920	1446×1840	1840×1446
8	4.93	5.06	3.00
60	32.81	31.61	21.62
120	54.30	52.61	41.38
240	82.19	78.32	75.74
244	82.27	78.84	76.69
2×244	140.50	153.45	155.52
4×244	215.37	305.00	313.83
6×244	263.83	420.93	436.77
8×244	294.13	511.25	543.02
10×244	305.02	583.50	622.05
12×244	328.53	646.32	669.49
16×244	346.29	724.19	811.22

Comparison


Execution time
CPU 723× 920
CPU 1446×1840
CPU 1840×1446
Phi 723× 920
Phi 1446×1840
Phi 1840×1446
CPU+Phi 723× 920
CPU+Phi 1446×1840
CPU+Phi 1840×1446
Time [s]
10^3
10^2
1
2
4
8
number of nodes