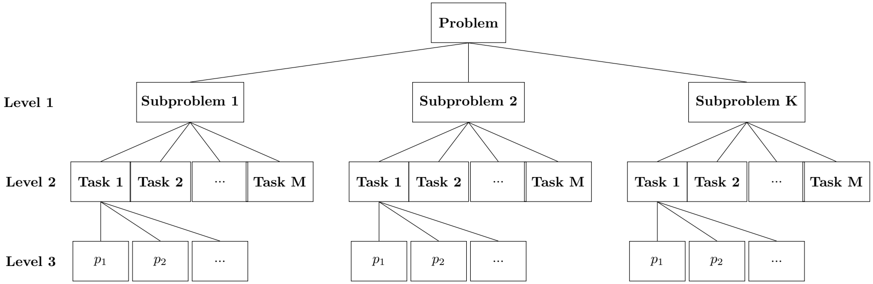
Vieno uždavinio trijų lygmenų lygiagretinimo schemos tyrimas

Rima Kriauzienė (VU DAMSTI, VGTU), Andrej Bugajev (VGTU), Raimondas Čiegis (VGTU)

Balandžio 13 d., 2018 m., Vilnius, Lietuva

The problem and parallelization strategies



As an example, it can be M different PDEs $L_j u = 0, 1 \le j \le M$, which are approximated numerically with $L_j^h U = 0, 1 \le j \le M$ with solutions u^h. Solutions depend on parameters $q_1, q_2, \dots, q_m$.

The optimization problem

$$\min_{q_1, q_2, \dots, q_m} F(u_1^h(q_1, q_2, \dots, q_m), \dots, u_M^h(q_1, q_2, \dots, q_m))$$

Balancing of workload distribution

- Let P be the number of parallel processes.
- We consider the following main minimisation problem: find the optimal value k_0 of task blocks

$$T_0(k_0) = \min_{1 \le k \le K} T_B\left(P/k\right)/\Gamma(k), \tag{1}$$

- $\Gamma(k) = k\gamma_k$, $0 < \gamma_k \le 1$, where $T_B(p)$ defines the optimal time for solving one block of M tasks using p processes:

Balancing of workload distribution

- Let P be the number of parallel processes.
- We consider the following main minimisation problem: find the optimal value k_0 of task blocks

$$T_0(k_0) = \min_{1 \le k \le K} T_B\left(P/k\right)/\Gamma(k), \tag{1}$$

- $\Gamma(k) = k\gamma_k$, $0 < \gamma_k \le 1$, where $T_B(p)$ defines the optimal time for solving one block of M tasks using p processes:

$$T_B(P) = \min_{(p_1,\dots,p_M)\in S} \max_{1 \le m \le M} t_m(p_m), \tag{2}$$

where a set S of feasible processors distributions is defined as

$$S = \left\{(p_1, \dots, p_M): \quad p_m \le P_m,\ m = 1, \dots, M, \quad p_1 + \dots + p_M \le P\right\}.$$

ALGORITHM 1: Distribution of P processes between M tasks

```
Set $p[m] = 1$, for $m = 1, \ldots, M$
$P = P - M$
Compute $t_m(p[m])$, for $m = 1, \ldots, M$
stop $= 0$
while $P > 0$ & $stop == 0$ do
    find $j$ such that $t_j(p[j]) = \max_{1 \le m \le M} t_m(p[m])$
    if $p[j] == P_j$ then
        stop $= 1$
    else
        $p[j] = p[j] + 1;\ P = P - 1;$
    end
end
```

In the presented algorithm P_j is calculated taking into account two restrictions:

- The number of processes cannot exceed the number after which the speed-up begin to drop.
- The number of processes is limited by efficiency requirement, which means it is not allowed increase the number of processes per task v_j if it makes the calculations efficiency smaller than the selected constant E_{min}.

Balancing of workload distribution

- Let $t_j(p)$ be the time that is needed to compute j–th task with p processes.

$$P_j = \min\,(\bar{P}_j, \tilde{P}_j). \tag{3}$$

- where $\bar{P}_j$ is a global minimum

$$t_j(p) \geq t_j(\bar{P}_j), \quad \text{for} \;\; p > \bar{P}_j, \tag{4}$$

- $\tilde{P}_j$ is the maximum number of processes which satisfies the efficiency condition

$$\frac{t_j(1)}{p\, t_j(p)} \geq E_{min}, \quad \text{for} \;\; p \leq \tilde{P}_j, \tag{5}$$

  where $E_{min} \in [0, 1]$ is a constant.

First level of parallel algorithm

As a local optimizer Nelder-Mead algorithm is used.

During each iteration we can have these different scenarios

- Reflection (one point: f_R)
- Expansion (two points: f_R, f_e)
- Contraction (two points: f_R, f_c)

Note: in this case $K = 3$

Linear Schrödinger equation

$$\begin{aligned} & i\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} = 0,\ x \in (a,b), t \in [0,T] \\ & u(x,0) = u_0(x) \\ & L_l u(a) = 0, L_r u(b) = 0. \end{aligned} \tag{6}$$

The boundary conditions approximation [1]

$$\partial_n u = -e^{-i\frac{\pi}{4}} \left(\left(\sum_{k=1}^{\frac{m+1}{2}} q_k \right) u - \sum_{k=1}^{\frac{m-1}{2}} q_{k+1} q_{k+(m+3)/2} \varphi_k \right), \tag{7}$$

where $\partial_n u$ is the normal derivative, the number of parameters $m \in \mathbb{N}$ is odd and φ_k is obtained from

$$\frac{d\varphi_k(x,t)}{dt} + q_{k+(m+3)/2}\varphi_k(x,t) = u\,(x,t)\,,\ x = a, b,\ k = 1, \dots, [m/2].$$

[1] A. Bugajev, R. Čiegis, R. Kriauzienė, T. Leonavičienė and J. Žilinskas (2017). On the Accuracy of Some Absorbing Boundary Conditions for the Schrödinger Equation. Mathematical Modelling and Analysis, 22(3):408-423.

Optimization problem

Then we formulate an optimization problem

$$\min_{q_1, q_2, \ldots, q_m} \max_{1 \le j \le M} \| u_j - u_j^h(q_1, q_2, \ldots, q_m) \|_\infty$$

Notes

- Assuming that u_j is known, each value of functional that is being minimized requires to solve M different equations.
- All M equations can be solved independently.

Second level of parallel algorithm

We compute functional

$$\max_{1 \le j \le M} \| u_j - u_j^h(q_1, q_2, \dots, q_m) \|_\infty = F(q_1, q_2, \dots, q_m), \qquad (8)$$

where u^h approximation error should be small comparing to F.

Tasks

U_j with different j can be computed independently, computation of maximum is small comparing to computations of finding solutions and computational errors. This leads to efficient parallel calculation of solutions.

Heterogenousity

Different PDEs can require different discretization sizes in order to achieve the same level of errors. This leads to unequal computational costs for different problems leading to loss of parallel efficiency.

Third level of parallel algorithm

The system of linear equations

$$\begin{cases} b_0 x_0 + c_0 x_1 = d_0, \\ a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \quad i = 1, \dots, N-2 \\ a_{N-1} x_{N-2} + b_{N-1} x_{N-1} = d_{N-1}, \end{cases} \tag{9}$$

where a_i, b_i, c_i, d_i are complex numbers

Linear equations with tridiagonal matrix are solved using Wang's algorithm.

Wang's algorithm complexity

The computational complexity for each parallel process is

$$T_{Wp} = 17\frac{J}{p} + 8p + T_{c1}(p), \tag{10}$$

where J is the size of system, p is the number of processes, where $T_{c1}(p)$ defines communication costs.

Improvement on the second level

Third level let us to perform the workload balancing at the second level of our algorithm. We assign different numbers of processes for different problems with different computational costs.

Workload distribution

Simple rule

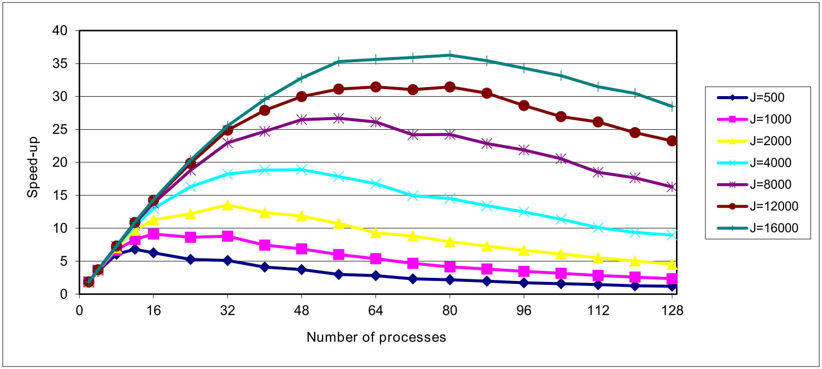
We can use a simple rule to choose the number of processes: it must be proportional to $Z_j = J_j \cdot N_j$, where J_j, N_j are discretization sizes in x and t directions respectively.

Table: Example of different sizes of problems and distribution of processes using simple rule

j	1	2	3	4
Z_j	3.2e+07	4.8e+07	1.152e+07	9.6e+07

p \ j	1	2	3	4
4	1	1	1	1
8	1	2	1	4
16	3	4	1	8
32	6	8	2	16
64	11	17	4	32
128	22	33	8	65

Workload distribution



For bigger number of processes p a simple rule of proportionality to the sizes of problems is not enough.

Solutions

Table: Benchmarks with different sizes

Benchmark 1		Benchmark 2		Benchmark 3	
Eq.	Sizes	Eq.	Sizes	Eq.	Sizes
1	8000×40000	1	8000×20000	1	8000×10000
2	4000×20000	2	4000×20000	2	2000×20000
3	2000×20000	3	4000×10000	3	2000×10000
4	2000×10000	4	2000×10000	4	1000×20000

The results of the first benchmark

p	16	32	64	96	128	128
		$k = 1$		$k = 2$	$k = 3$	$K = 1$
Eq. 1	10	22	50	34	29	56
Eq. 2	3	5	8	8	7	8
Eq. 3	2	3	4	4	4	4
Eq. 4	1	2	2	2	2	2
Model time	11.145	5.784	3.614	2.742	2.272	3.605
Exp. time	11.003	5.394	3.608	2.719	2.308	3.600
Speed-up	12.679	25.862	38.664	51.307	60.444	38.75

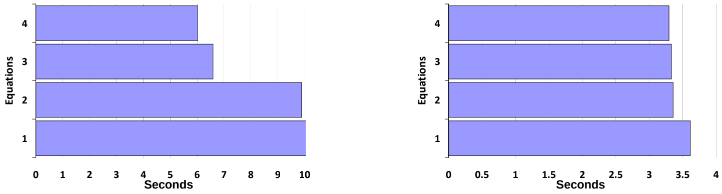


Figure: Benchmark 1 T values with $p = 16$(left) and $p = 64$(right)

The results of the second and third benchmarks

Table: The results of the second benchmark

p	16	32	64	96	128	128
	$k = 1$			$k = 2$		$K = 1$
Eq. 1	9	18	37	26	37	56
Eq. 2	4	8	15	12	15	18
Eq. 3	2	4	8	7	8	8
Eq. 4	1	2	4	3	4	4
Model time	6.59	3.36	2.01	1.65	1.34	1.8
Exp. time	6.69	3.37	1.98	1.62	1.33	1.86
Speed-up	13.6	27.03	46.03	56.24	68.25	49.03

Table: The results of the third benchmark

p	16	32	64	96	128	128
	$k = 1$			$k = 2$		$K = 1$
Eq. 1	8	16	32	24	32	56
Eq. 2	4	8	16	12	16	31
Eq. 3	2	4	8	6	8	8
Eq. 4	2	4	8	6	8	9
Model time	3.33	1.76	1.05	0.87	0.7	0.9
Exp. time	3.38	1.76	1.06	0.86	0.7	0.95
Speed-up	14.33	27.55	45.96	56.72	69.08	51.23

The results of benchmark 1 with $E_{min} > 0$

p	128		
E_{min}	0.75	0.8	0.6
	$k = 3$	$k = 3$	$K = 1$
Eq. 1	26	19	42
Eq. 2	7	5	8
Eq. 3	4	3	4
Eq. 4	2	1	2
Model time	2.45478	3.167	3.8413
Exp. time	2.4899	3.075	3.75898
Speed-up	56.0289	45.373	37.112461

Benchmark 4

Problem 1

$$u_1(t,x) = \frac{\exp(-i\pi/4)}{\sqrt{4t-i}} \exp\left(\frac{ix^2 - 6x - 36t}{4t - i}\right).$$

$x \in [-5, 5]$, $t \in [0, 0.8]$. $J \times N = 8000 \times 4000$.

Problem 2

$$u_2(t,x) = \frac{1}{\sqrt[+]{1 + it/\alpha}} \exp\left(ik(x - x^{(0)} - kt) - \frac{(x - x^{(0)} - 2kt)^2}{4(\alpha + it)}\right),$$

where $k = 100, \alpha = 1/120, x^{(0)} = 0.8$. $x \in [0, 1.5]$, $t \in [0, 0.04]$,
$J \times N = 12000 \times 4000$

Benchmark 4

Problem 3

$$u_1(t,x) = \frac{\exp(-i\pi/4)}{\sqrt{4t-i}} \exp\left(\frac{ix^2 - 6x - 36t}{4t-i}\right).$$

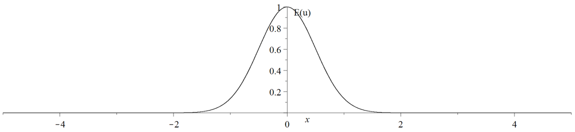
$x \in [-3,3]$, $t \in [0, 0.48]$. $J \times N = 16000 \times 10000$.

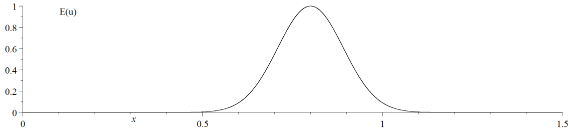
Problem 4

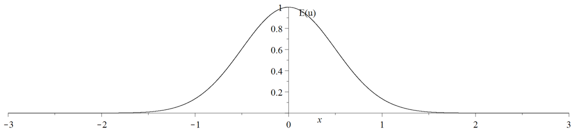
$$u_2(t,x) = \frac{1}{\sqrt[+]{1+it/\alpha}} \exp\left(ik(x - x^{(0)} - kt) - \frac{(x - x^{(0)} - 2kt)^2}{4(\alpha + it)}\right),$$

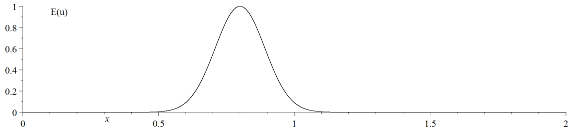
where $k = 100, \alpha = 1/120, x^{(0)} = 0.8$. $x \in [0, 1.5]$, $t \in [0, 0.06]$,
$J \times N = 16000 \times 8000$

Solutions of benchmark 4









Conclusions

1. Comparing to one level parallelization the proposed algorithm with three levels greatly expands the number of processes that can be used.

2. Workload balancing was analysed, model-based approach performs balancing well enough for practical purposes. The model prediction times are close to times of the real computational experiments.

3. We propose the heuristic with parameter E_{min} which guarantees that the efficiency of calculations on the third level is not lower than the value of E_{min}. This heuristic is not optimal, however, for considered cases we show that it is sufficient.

4. The number of processes scales well as the number of differential problems M raises, making it possible to perform computations even with big M.

Thank you for your attention