Sprawozdanie z OpenMP - podstawy

Patryk Wojtyczek

Program został zaimplementowany w c++ i wykorzystany został generator liczb losowych z biblioteki standardowej. Każdy wątek miał własny generator.

Testy zostały przeprowadzone na moim prywatnym laptopie - MacOS Intel core i9 z 8 fizycznymi rdzeniami.

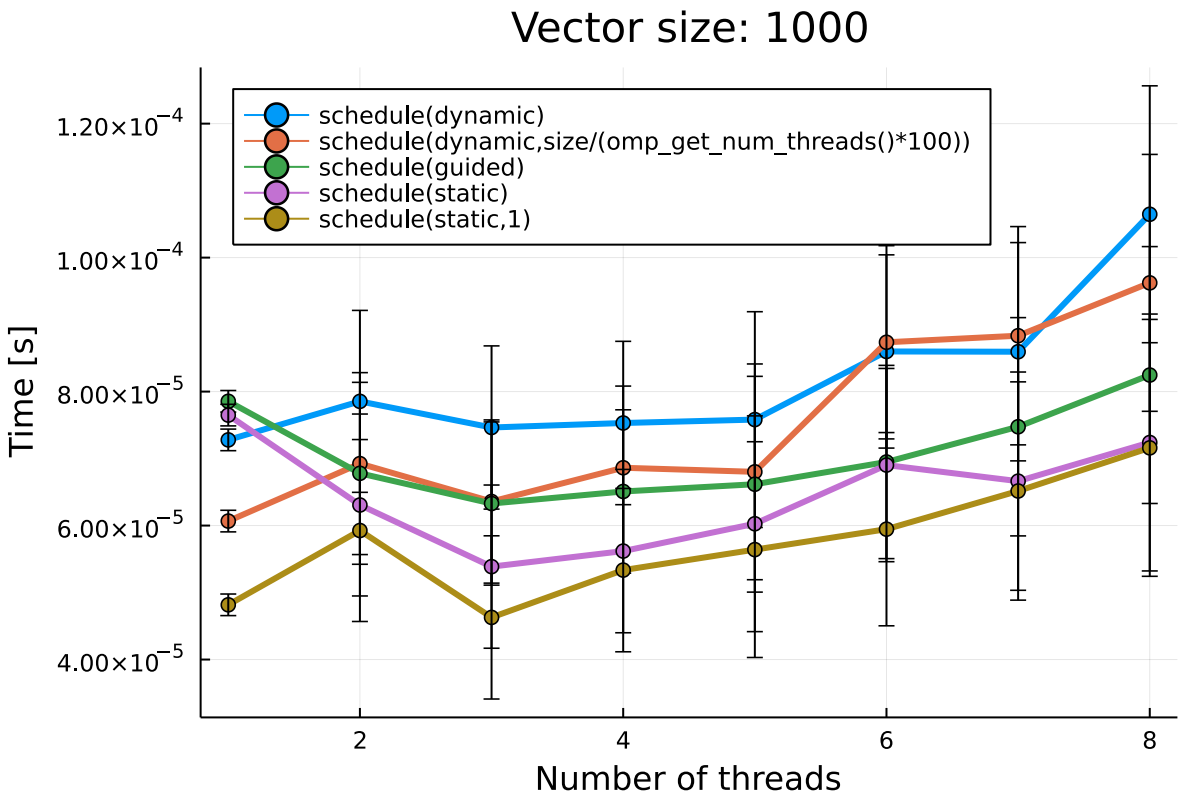
	threads	size	schedule	time
1	1	1000	"schedule(static)"	7.4e-5
2	1	1000	"schedule(static)"	7.7e-5
3	1	1000	"schedule(static)"	7.7e-5
4	1	1000	"schedule(static)"	7.7e-5
5	1	1000	"schedule(static)"	7.7e-5
6	1	1000	"schedule(static)"	7.7e-5
7	1	1000	"schedule(static)"	7.7e-5
8	1	1000	"schedule(static)"	7.7e-5
9	1	1000	"schedule(static)"	7.6e-5
10	1	1000	"schedule(static)"	7.6e-5
more				
12000	8	1000000	"schedule(guided)"	0.010191

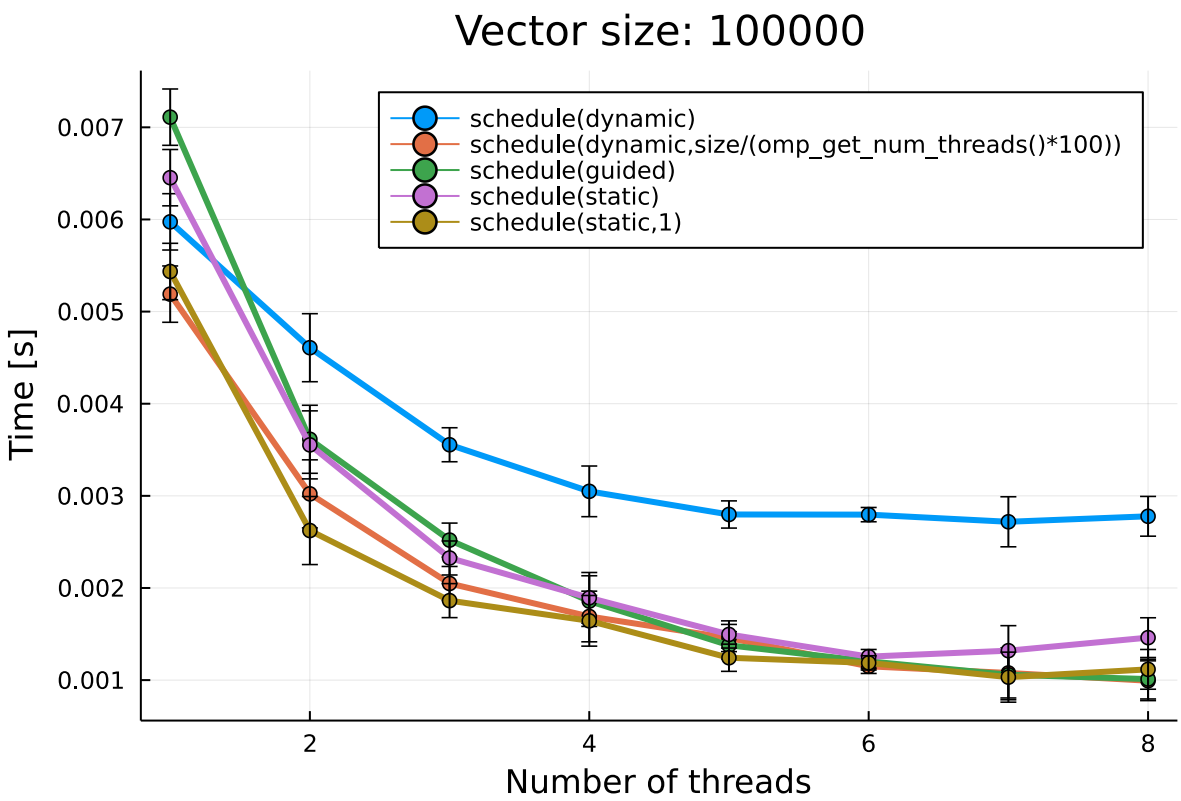
	threads	schedule	size	time
1	1	"schedule(static)"	1000	$7.65e-5 \pm 1.6e-6$
2	2	"schedule(static)"	1000	$6.3e-5 \pm 1.4e-5$
3	3	"schedule(static)"	1000	$5.4e-5 \pm 1.2e-5$
4	4	"schedule(static)"	1000	$5.6e-5 \pm 1.2e-5$
5	5	"schedule(static)"	1000	$6.0e-5 \pm 1.6e-5$
6	6	"schedule(static)"	1000	$6.9e-5 \pm 1.4e-5$
7	7	"schedule(static)"	1000	$6.7e-5 \pm 1.6e-5$
8	8	"schedule(static)"	1000	$7.2e-5 \pm 1.9e-5$

	threads	schedule	size	time
1	1	"schedule(static)"	100000	0.00645 ± 0.00031
2	2	"schedule(static)"	100000	0.00355 ± 0.00037
3	3	"schedule(static)"	100000	0.00233 ± 0.00018
4	4	"schedule(static)"	100000	0.00189 ± 0.00027
5	5	"schedule(static)"	100000	0.0015 ± 0.00015
6	6	"schedule(static)"	100000	$0.001255 \pm 7.8e-5$
7	7	"schedule(static)"	100000	0.00132 ± 0.00027
8	8	"schedule(static)"	100000	0.00146 ± 0.00022

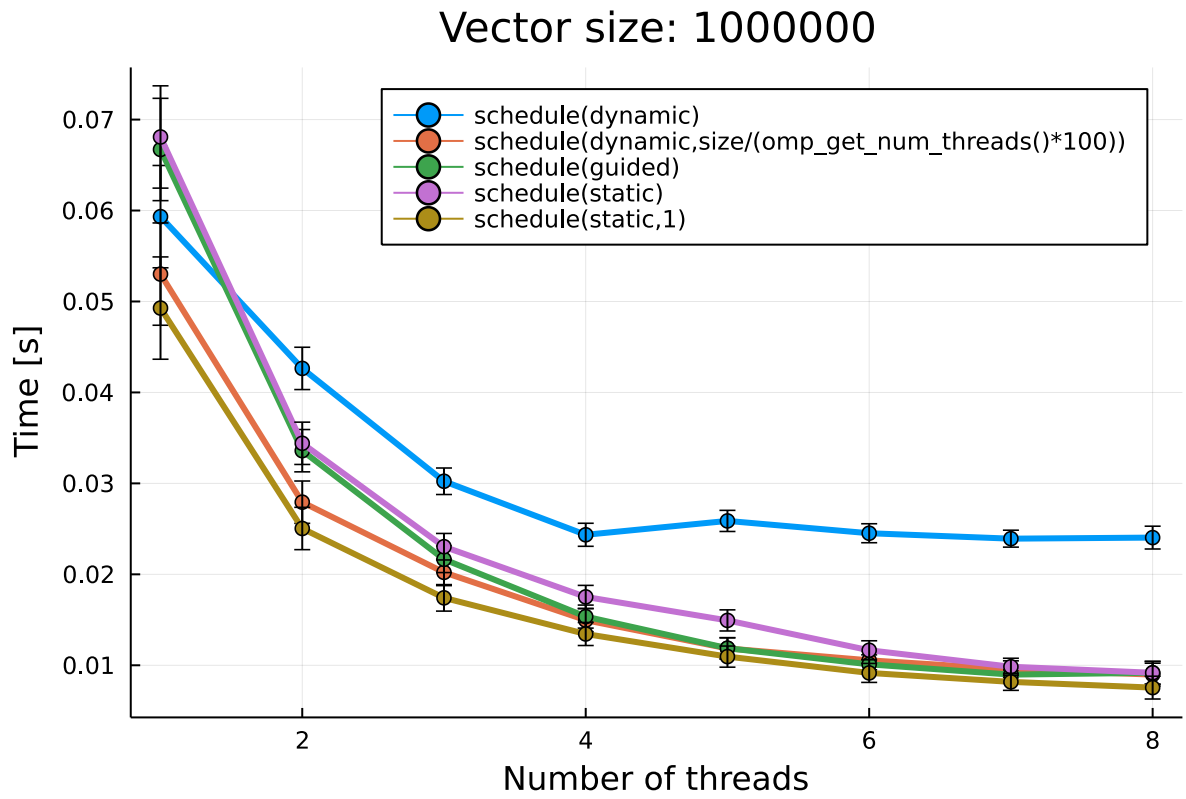
```
begin
    small_size_group, medium_size_group, big_size_group = groupby(groupped, :size)
    medium_size_group
end
```

Czas wykonania



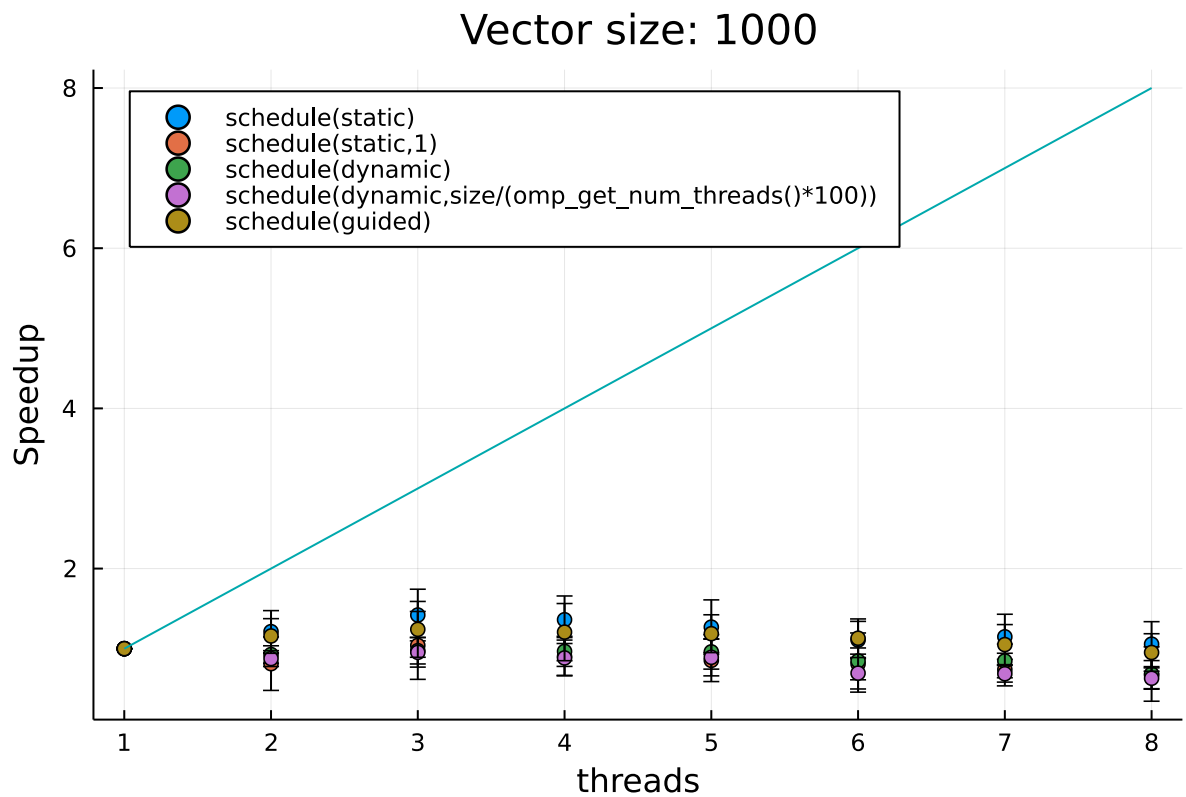


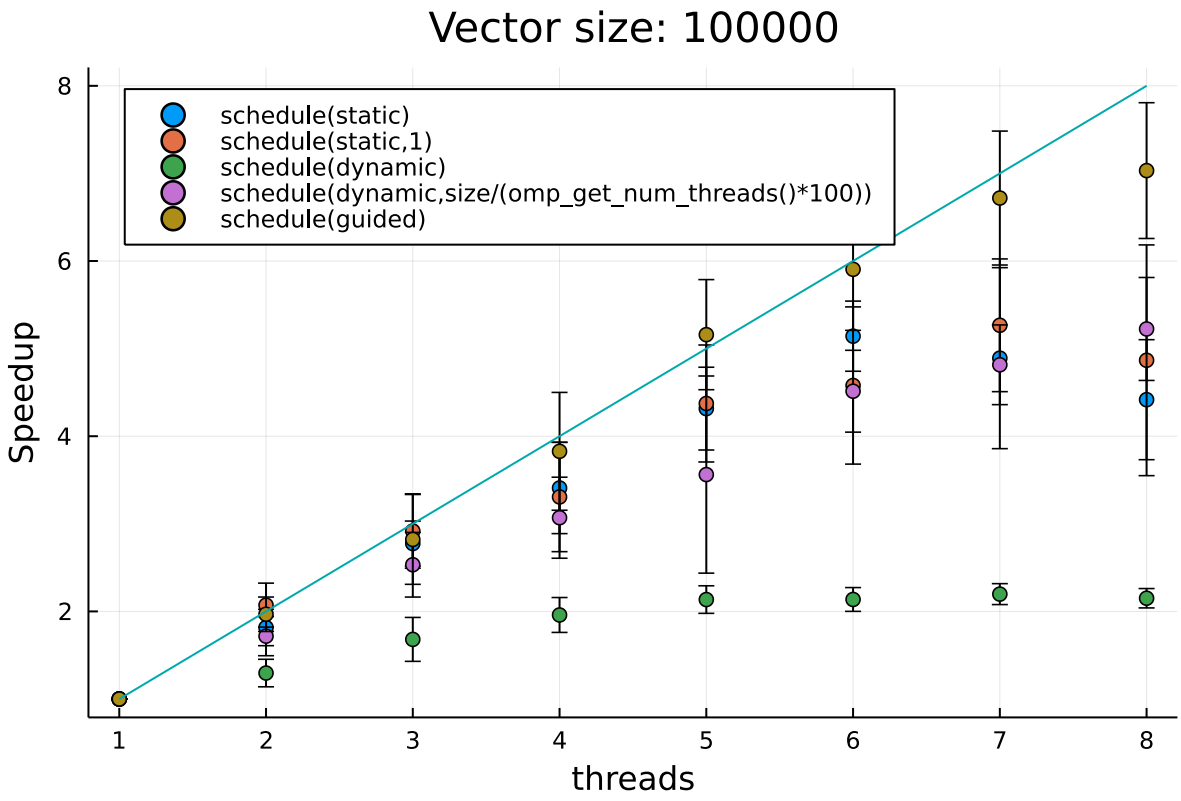

```
begin
    med_size = medium_size_group[1, :size]
    @df medium_size_group plot(:threads, :time, group=:schedule,
        xlabel = "Number of threads",
        ylabel = "Time [s]",
        title = "Vector size: $med_size",
        lw=3,
        marker=:circle)
end
```

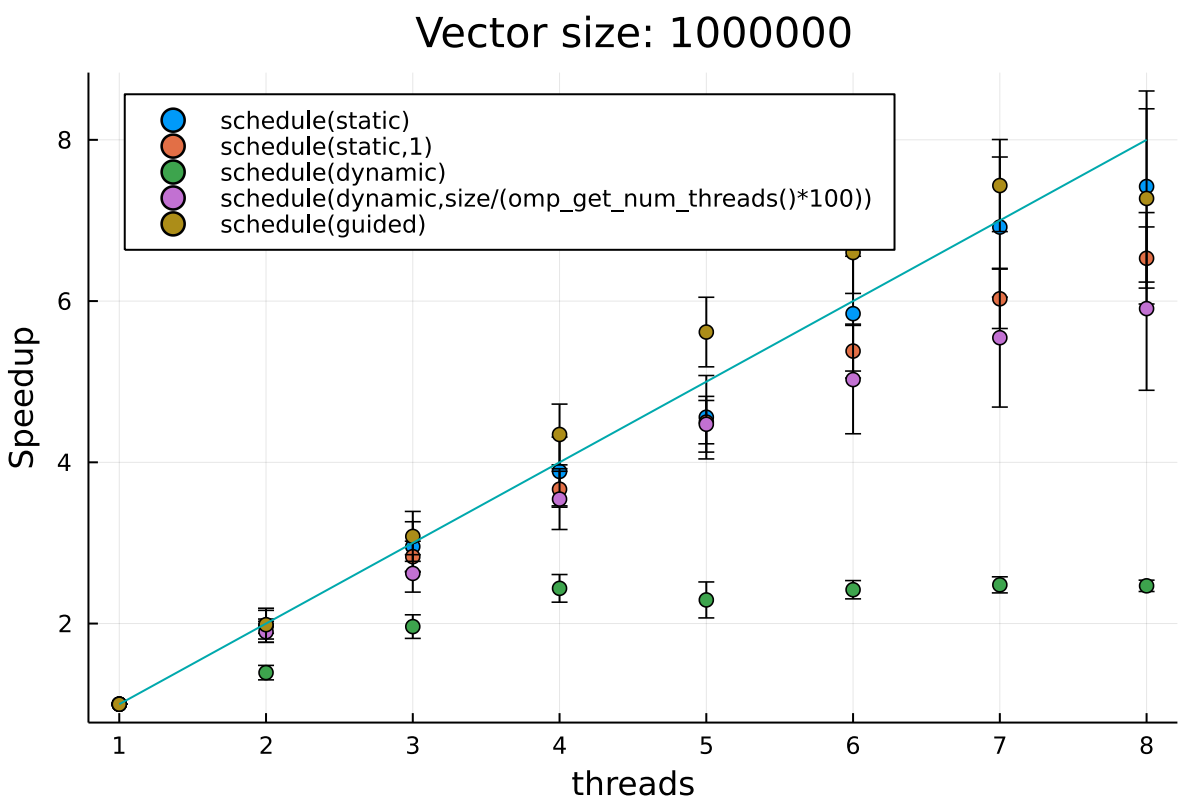


Speedup

```
plot_speedup! (generic function with 1 method)
```




Vector size: 100000
schedule(static)
schedule(static,1)
schedule(dynamic)
schedule(dynamic,size/(omp_get_num_threads()*100))
schedule(guided)
Speedup
threads
1
2
3
4
5
6
7
8
2
4
6
8


Vector size: 1000000
schedule(static)
schedule(static,1)
schedule(dynamic)
schedule(dynamic,size/(omp_get_num_threads()*100))
schedule(guided)
Speedup
threads
1
2
3
4
5
6
7
8
2
4
6
8

Wnioski

DLa małego rozmiaru wektora jakiekolwiek zrównoleglanie nie ma najmniejszego sensu co widać na wykresach - koszty synchronizacji/zarządzania wątkami w całości zjadają korzyści.

Właściwie to jednym mocno odstającym schedulerem był 'dynamic' z rozmiarem chunka 1. Scheduler ten działa na zasadzie przetwarzania chunka i proszenia o kolejne aż się skończą. Takie zachowanie ma sens dla problemów w których przetwarzanie chunków jest niezbalansowane - w naszym przypadku tak nie jest bo wykonujemy podobną pracę przy każdej iteracji. Koszt synchronizacji biorącej się z proszenia o chunki sprawia, że ten scheduler nie skaluje się tak dobrze.

Static radzi sobie bardzo dobrze dla tego problemu gdyż po prostu dzeli tablicę na chunki mniej więcej równej wielkości i daje je do przetwarzania każdemu wątkowi. Takie scheduling działa bardzo dobrze dla zbalansowanego przetwarzania tak jak tutaj.

Guided to swego rodzaju mix pomiędzy static a dynamic gdyż zasada jest podobna do dynamic ale rozmiary chunków zmniejszą się zamiast być stałe. Ten typ scheduling również dobrze sobie tutaj radzi z tych samych powodów co static.