TUGAS AKHIR
PARAREL ALGORITHM

M.IRFANSYAH (2106701255)

Start Slide

M.Irfansyah 2106701255

Permasalahan

Bila diberikan 2 buat vector ***A*** dan ***B*** yang masing-masing berukuran n dengan elemen masing-masing berupa bilangan integer (dapat ditentukan secara random), buatlah program parallel untuk menentukan hasil penjumlahan vector ***A*** + ***B***

M.Irfansyah 2106701255

Algoritma

```
import multiprocessing as mp
import numpy as np
import matplotlib.pyplot as plt
import time
```

```
#Mendefinisikan fungsi penjumlahan vektor
def vector_addition(A,B): #Pendefinisian penjumlahan vektor
  result = A+B #Misalkn penjumalahn vektor dengan suatu variabe
  time.sleep(0.1) #Melakukan penjedaan selama 0,1 sekon
  return result

#Meminta input banyaknya elemen vektor
n=
```

```
#Menggenerasi vektor dengan input random
A = np.random.randint(-1000, 1000, n) # memilih random dari -1000 sampai 1000
B = np.random.randint(-1000, 1000, n) # memilih random dari -1000 sampai 1000


#Proses paralel
if __name__ == '__main__':
  running_time = [] #Mendefinikan bahwa variabel runingg time adalah sebuah list kosong
  NumPar = [2,4,6,8,10,12,14,16] # Memilih jumlah cores
  input= [] #Mendefinikan bahwa variabel input adalah sebuah list kosong
```

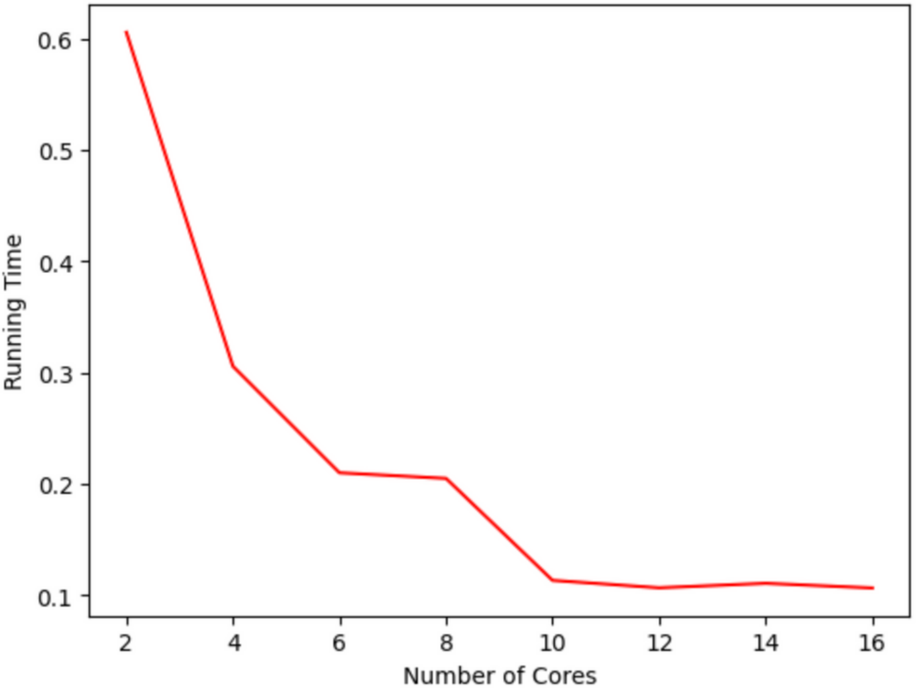
```
for i in range(n):
  input.append((A[i],B[i])) # input akan bertambah sepanjang n

for p in NumPar:
  pool = mp.Pool(p)
  t = time.time()
  result = pool.starmap(vector_addition, input)
  t2 = time.time()
  running_time.append(t2 - t)

#plotting running time
plt.plot(NumPar, running_time, "r-")
plt.xlabel('Number of Cores')
plt.ylabel('Running Time')
plt.show()
```

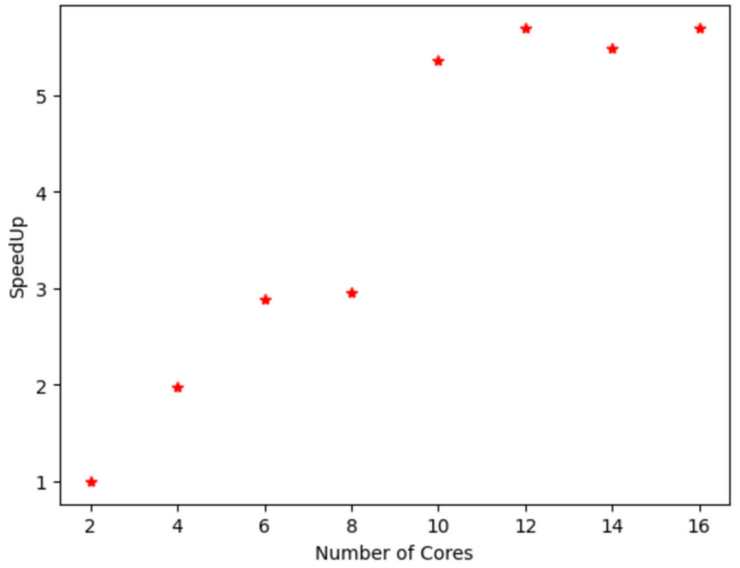
```
#Menghitung nilai speed-up beserta plottingnya
S = []
E = []
for m in range(len(NumPar)):
  S.append(running_time[0]/running_time[m]) # Menghitung S menggunakan T sequence dibagi dengan T par dengan m itu dari 0

plt.plot(NumPar, S, 'r*')
plt.xlabel('Number of Cores')
plt.ylabel('SpeedUp')
plt.show()
```

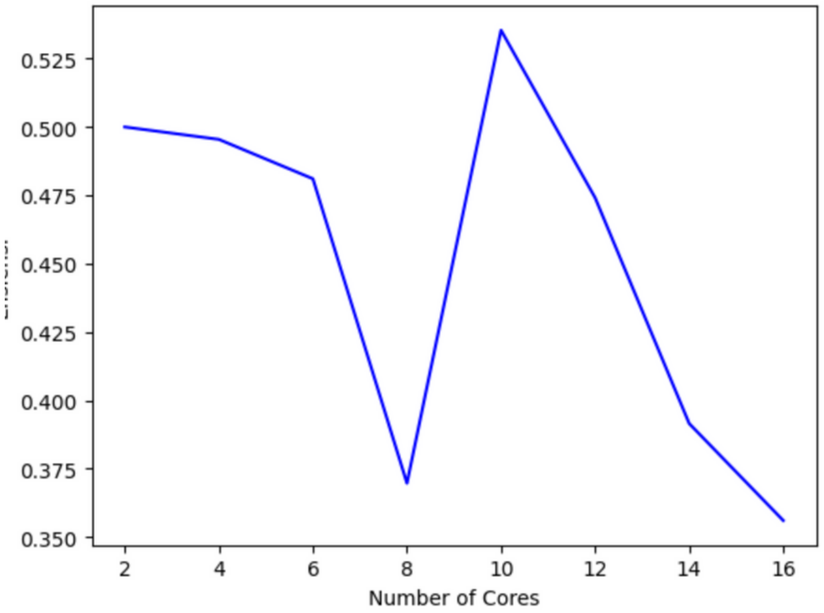
```
#Menghitung nilai efisiensi beserta plottingnya
for l in range(len(NumPar)):
  E.append(S[l]/NumPar[l]) # Menghitung E menggunakan S sequence dibagi dengan numpar l dengan l dimulai 0

plt.plot(NumPar, E, 'b-')
plt.xlabel('Number of Cores')
plt.ylabel('Efisiensi')
plt.show()
```

M.Irfansyah 2106701255

OUTPUT(n=10)

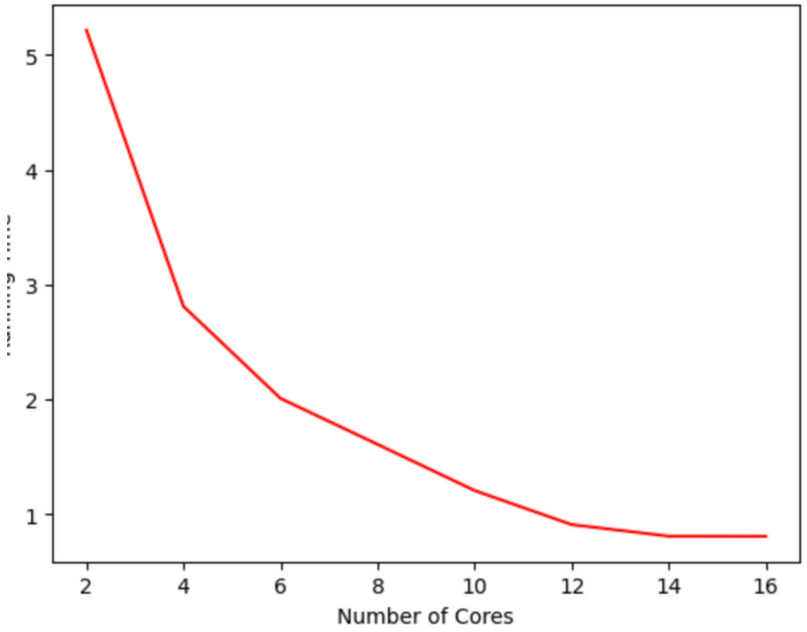


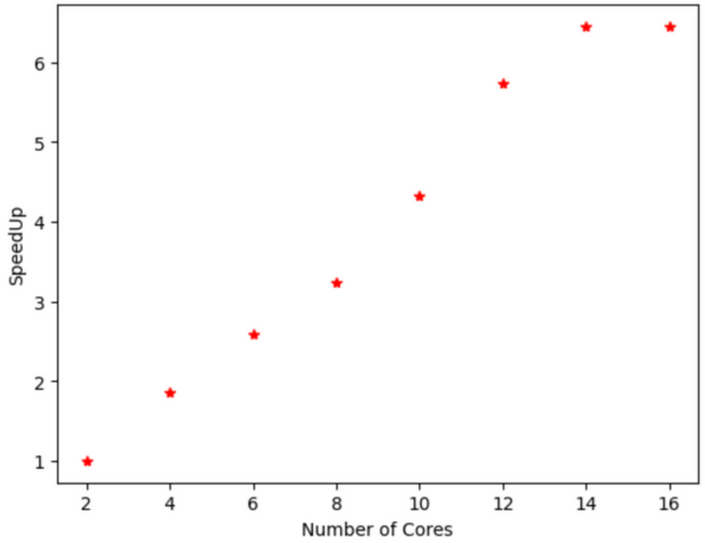


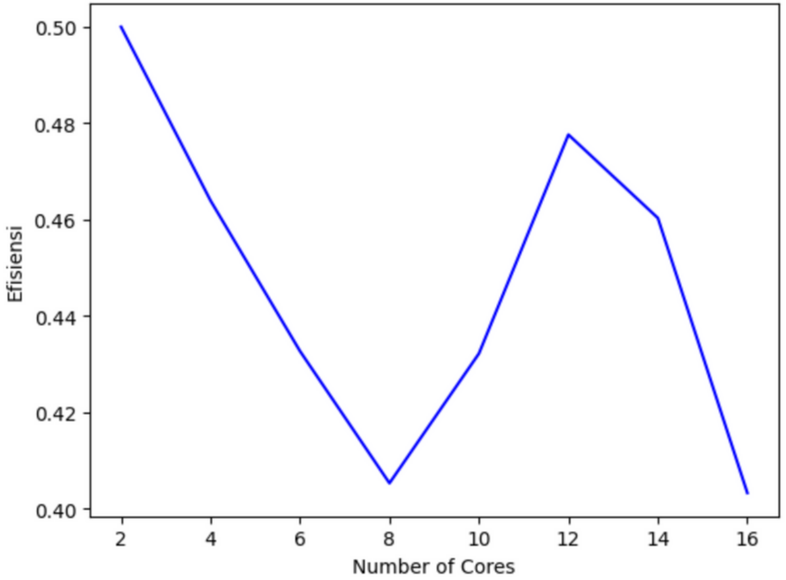


M.Irfansyah 2106701255

OUTPUT(n=100)

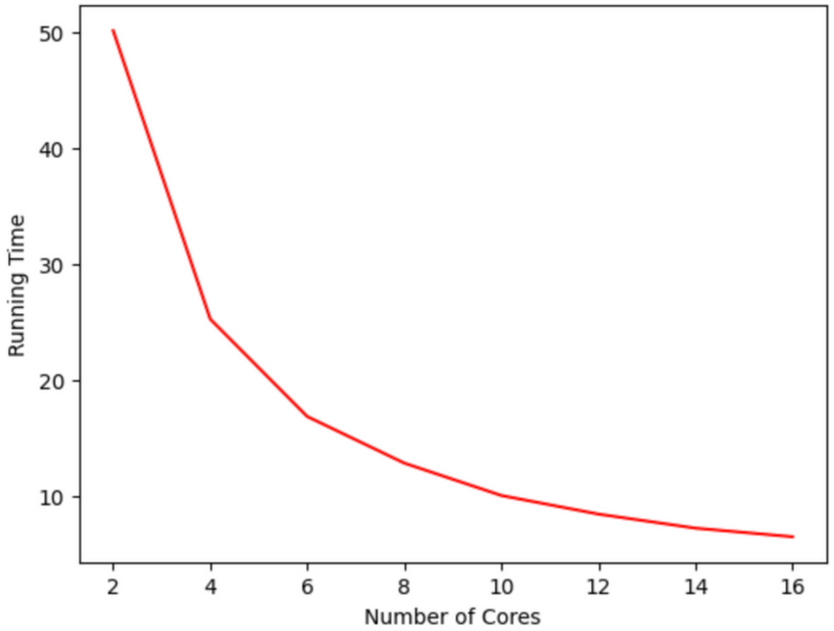


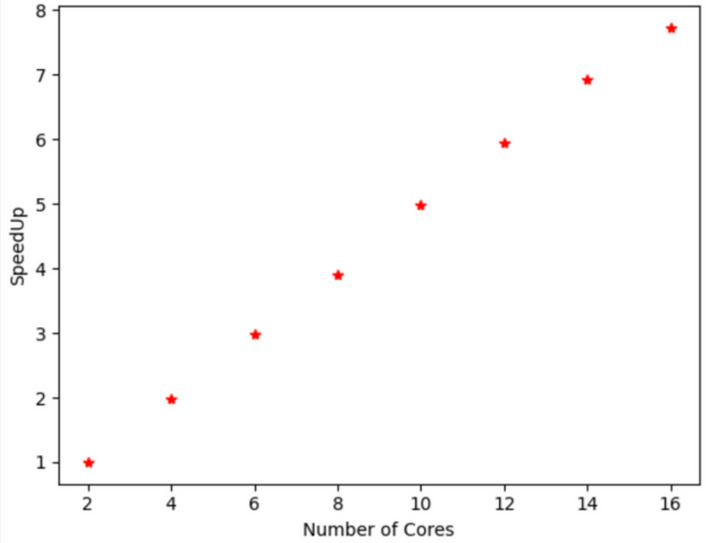


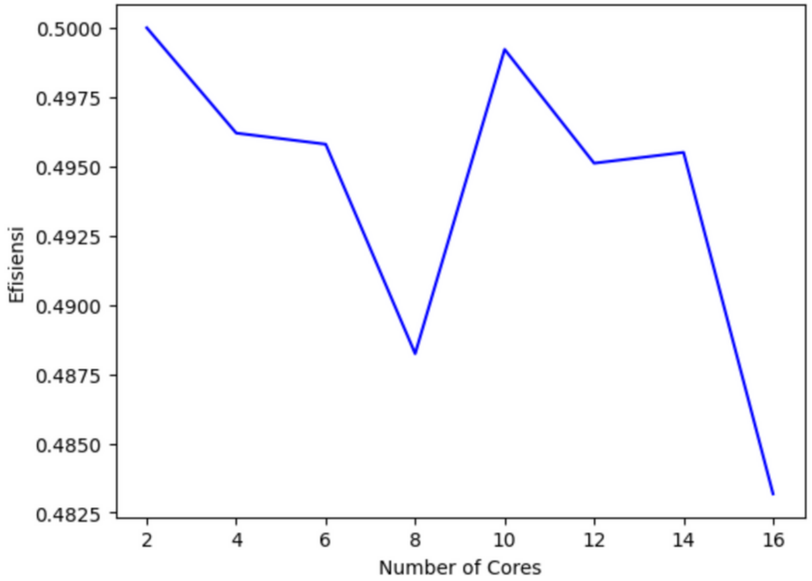


M.Irfansyah 2106701255

OUTPUT(n=1000)







M.Irfansyah 2106701255

OUTPUT

Cores	*Running Time*	*Speed Up*	*Efisiensi*
2	0.6061515808105469	1.0	0.5
4	0.30478358268737793	1.9887934102811817	0.49719835257029543
6	0.20817184448242188	2.91178464752341	0.4852974412539017
8	0.2040271759033203	2.9709355046789256	0.3713669380848657
10	0.11234283447265625	5.395551782682513	0.5395551782682513
12	0.11182498931884766	5.420537793052866	0.45171148275440554
14	0.11441826820373535	5.297681832862754	0.3784058452044824
16	0.10731649398803711	5.648261122638952	0.3530163201649345

N=10

Cores	*Running Time*	*Speed Up*	*Efisiensi*
2	5.215001821517944	1.0	0.5
4	2.8102569580078125	1.8557028412145102	0.46392571030362756
6	2.008814811706543	2.5960590250166753	0.43267650416944586
8	1.60850191116333	3.242148352653343	0.40526854408166785
10	1.2066762447357178	4.321790409207995	0.43217904092079945
12	0.9098958969116211	5.731426901933245	0.4776189084944371
14	0.8092403411865234	6.444317659536857	0.46030840425263264
16	0.8082981109619141	6.451829777644585	0.4032393611027866

N=100

Cores	*Running Time*	*Speed Up*	*Efisiensi*
2	50.12550210952759	1.0	0.5
4	25.266677141189575	1.9838581001145306	0.49596452502863264
6	16.848427772521973	2.975084843897248	0.49584747398287465
8	12.836102962493896	3.905040513930937	0.4881300642413671
10	10.034436464309692	4.995347998645773	0.4995347998645773
12	8.435884237289429	5.941938118111686	0.495161509842640 5
14	7.238030672073364	6.92529561983314	0.4946639728452243
16	6.427648067474365	7.798420446068938	0.48740127787930865

N=1000

M.Irfansyah 2106701255

KESIMPULAN

- Efisiensi untuk algoritma ini masih terbilang normal karena berkisaran antara 0 sampai dengan 1
- Speed up untuk algoritma ini berkisaran antara 1 hingga 7,8 yang mana ini mennjukkan bahwa speed up Normal

THANK YOU

End Slide