Backtesting Arbitrage-Strategy Trading System

Group 26

0516075王威堯　0516094張維哲　0516045張凱翔

OUTLINE

01 INTRODUCTION

02 PROBLEM STATEMENT

03 PROPOSED SOLUTION

04 EVALUATION

05 RELATED WORK

06 CONCLUSION

01

Introduction

Introduction

- High frequency trading is becoming important in financial markets
- Arbitrage opportunity comes from occasionally irrational price quotes.
- Parallel Programming is very suitable for finding arbitrage opportunity in big data financial markets.

Introduction

- Convexity Strategy Formula(Merton, 1973)：

$(X_3 - X_2) * C_{x1} + (X_2 - X_1) * C_{x3} - (X_3 - X_1) * C_{x2} \geq C_s$

X_1 、X_2、X_3 ：exercise price

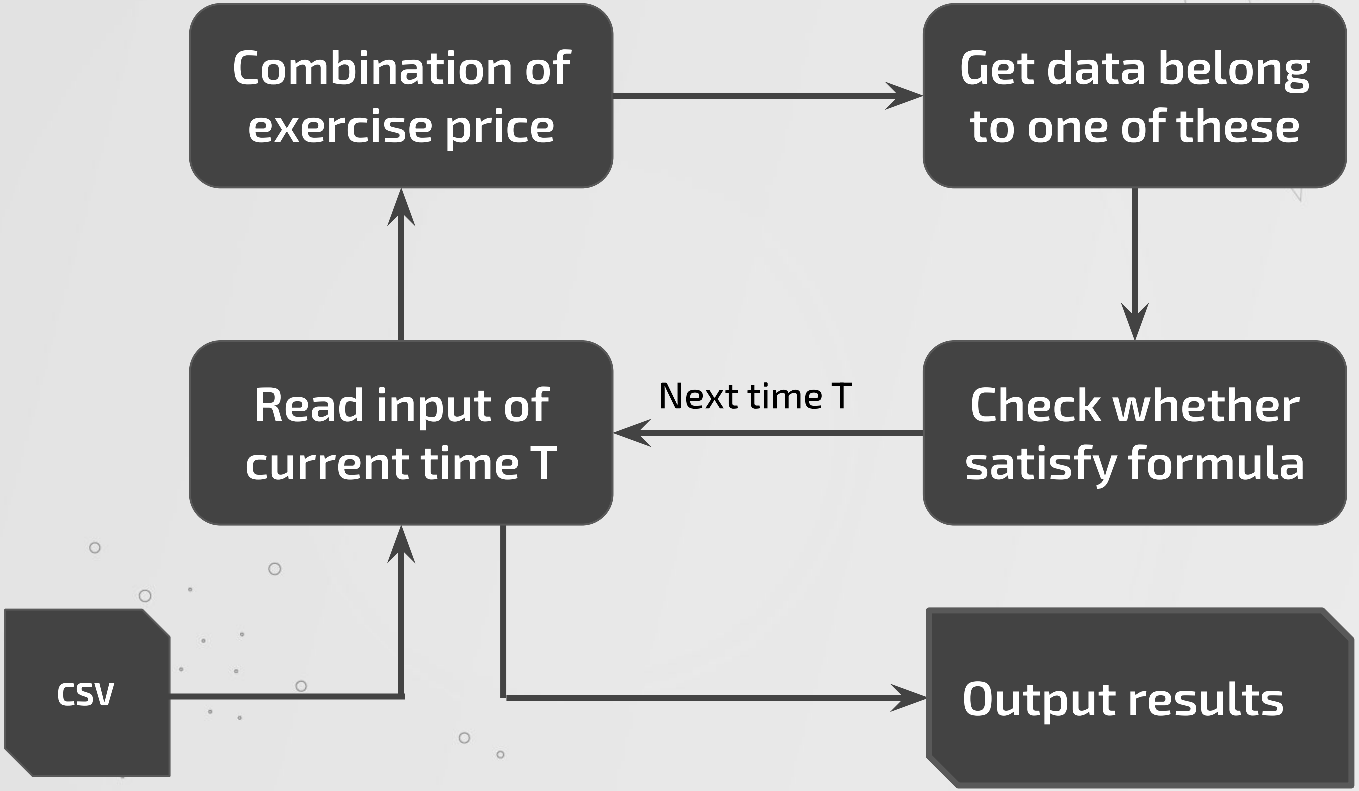
C_{x1} 、C_{x2} 、C_{x3} ：the price of exercise price

C_s：trading cost

02

Problem Statement

Flow Chart



Combination of exercise price

Get data belong to one of these

Check whether satisfy formula

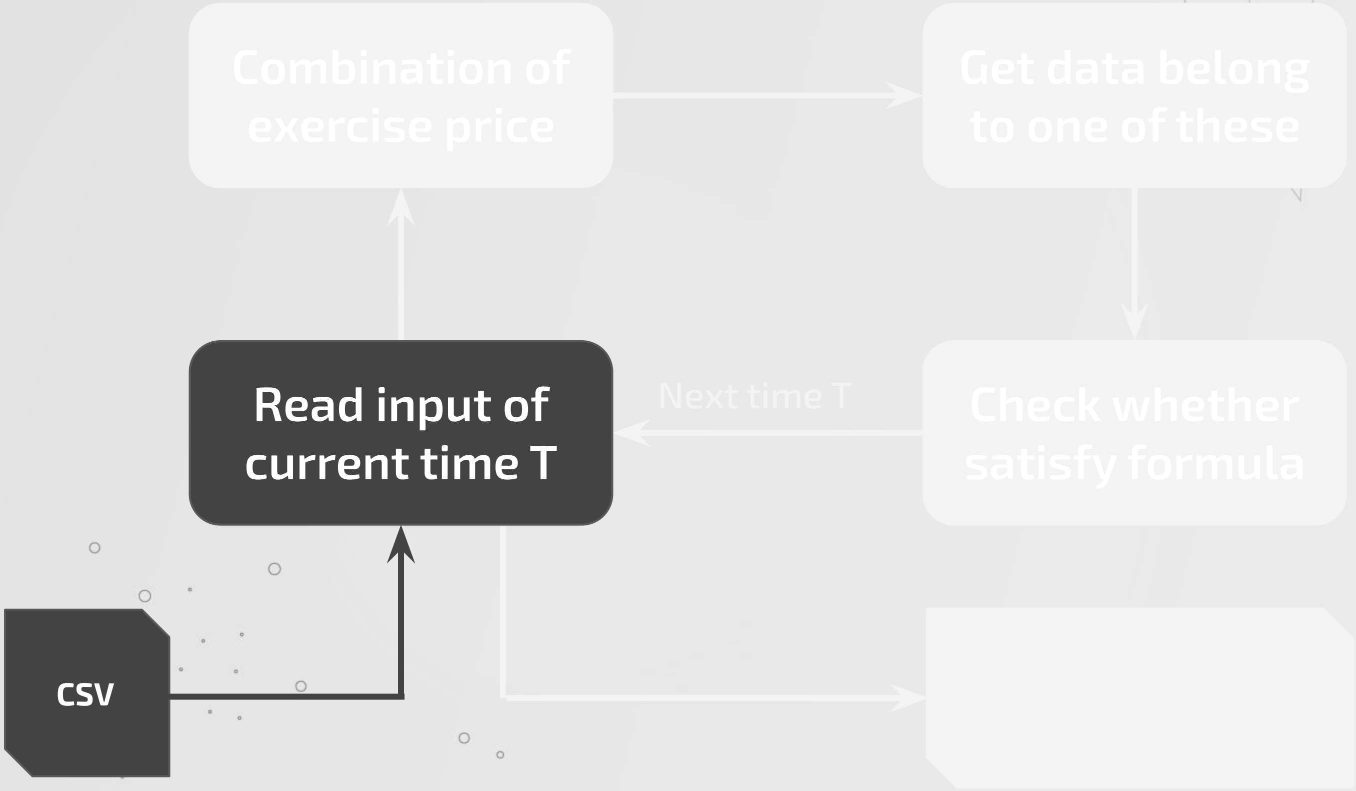
Next time T

Read input of current time T

csv

Output results

Example

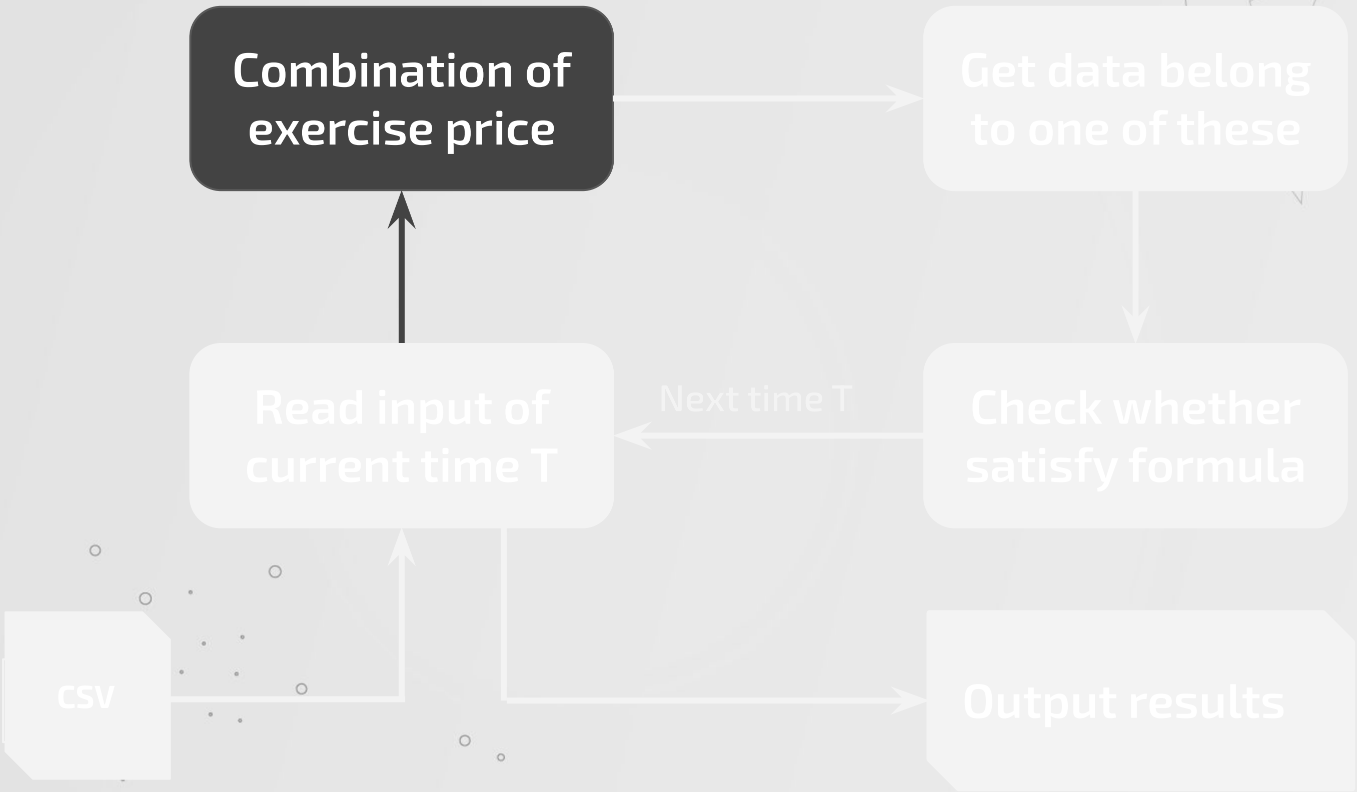

Combination of exercise price
Get data belong to one of these
Read input of current time T
Next time T
Check whether satisfy formula
csv

Example

- CSV

Exercise Price	Put/Call	Time	Price	Volume
8200	C	84500	101	2
8250	C	84500	101	2
8250	C	84500	69	6
8250	C	84500	69	6
8250	C	84500	69	5
8300	C	84500	69	5
8300	C	84500	40.5	36
8400	C	84500	40.5	36

Example



Combination of exercise price

Get data belong to one of these

Check whether satisfy formula

Next time T

Read input of current time T

csv

Output results

Example

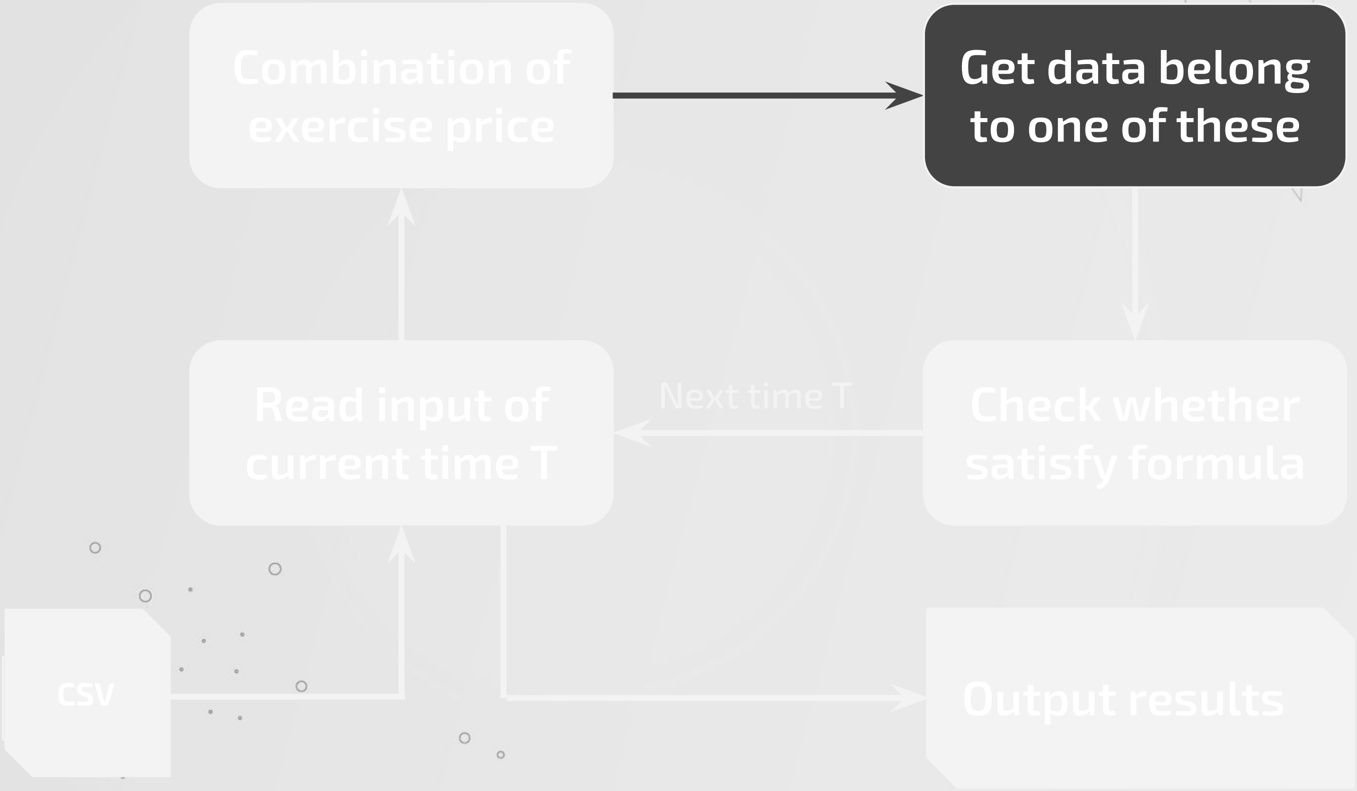
- Find unique exercise price

8200	8250	8300	8400

- Get combination of the exercise price (C(n,3))

8200	8250	8300
8200	8250	8400
8200	8300	8400
8250	8300	8400

Example



Combination of exercise price

Get data belong to one of these

Read input of current time T

Next time T

Check whether satisfy formula

csv

Output results

Problem Statement: Example

- Get data belong to certain exercise price
  - 8200

8200	C	84500	101	2

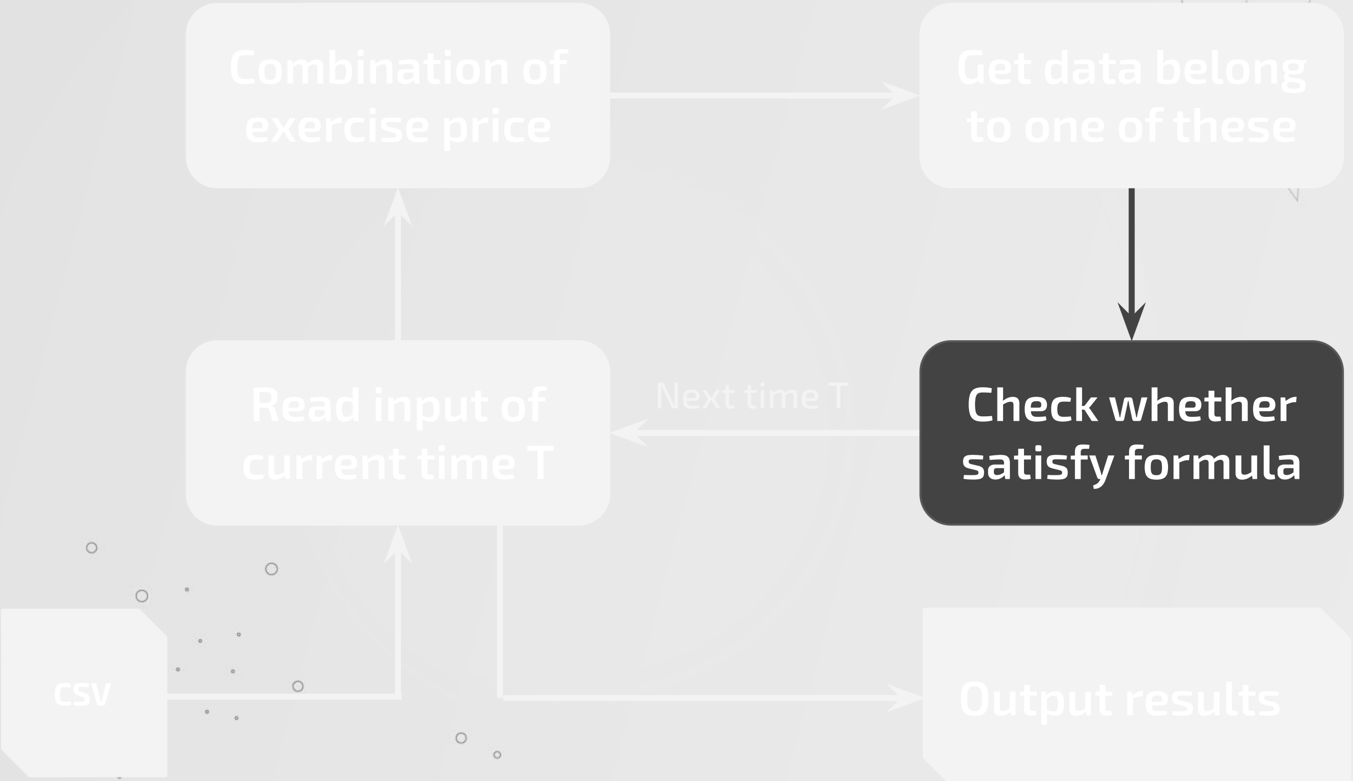
  - 8250

8250	C	84500	101	2
8250	C	84500	69	6
8250	C	84500	69	6
8250	C	84500	69	5

  - 8300

8300	C	84500	69	5
8300	C	84500	40.5	36

Flow Chart



Combination of exercise price

Get data belong to one of these

Check whether satisfy formula

Next time T

Read input of current time T

csv

Output results

Problem Statement: Example

- Get data belong to certain exercise price
  - 8200

8200	C	84500	101	2

  - 8250

8250	C	84500	101	2
8250	C	84500	69	6
8250	C	84500	69	6
8250	C	84500	69	5

  - 8300

8300	C	84500	69	5
8300	C	84500	40.5	36

Problem Statement: Example

- Get one data from each exercise price

8200	C	84500	101	2
8250	C	84500	101	2
8300	C	84500	69	5

- Check whether the equation is satisfied
  - $(X_3 - X_2)*C_{x1} + (X_2 - X_1)*C_{x3} - (X_3 - X_1)*C_{x2} \geq C_s$
- If TRUE then count + 1
- Return the answer and continue to do the next timestamp

03

Proposed Solution

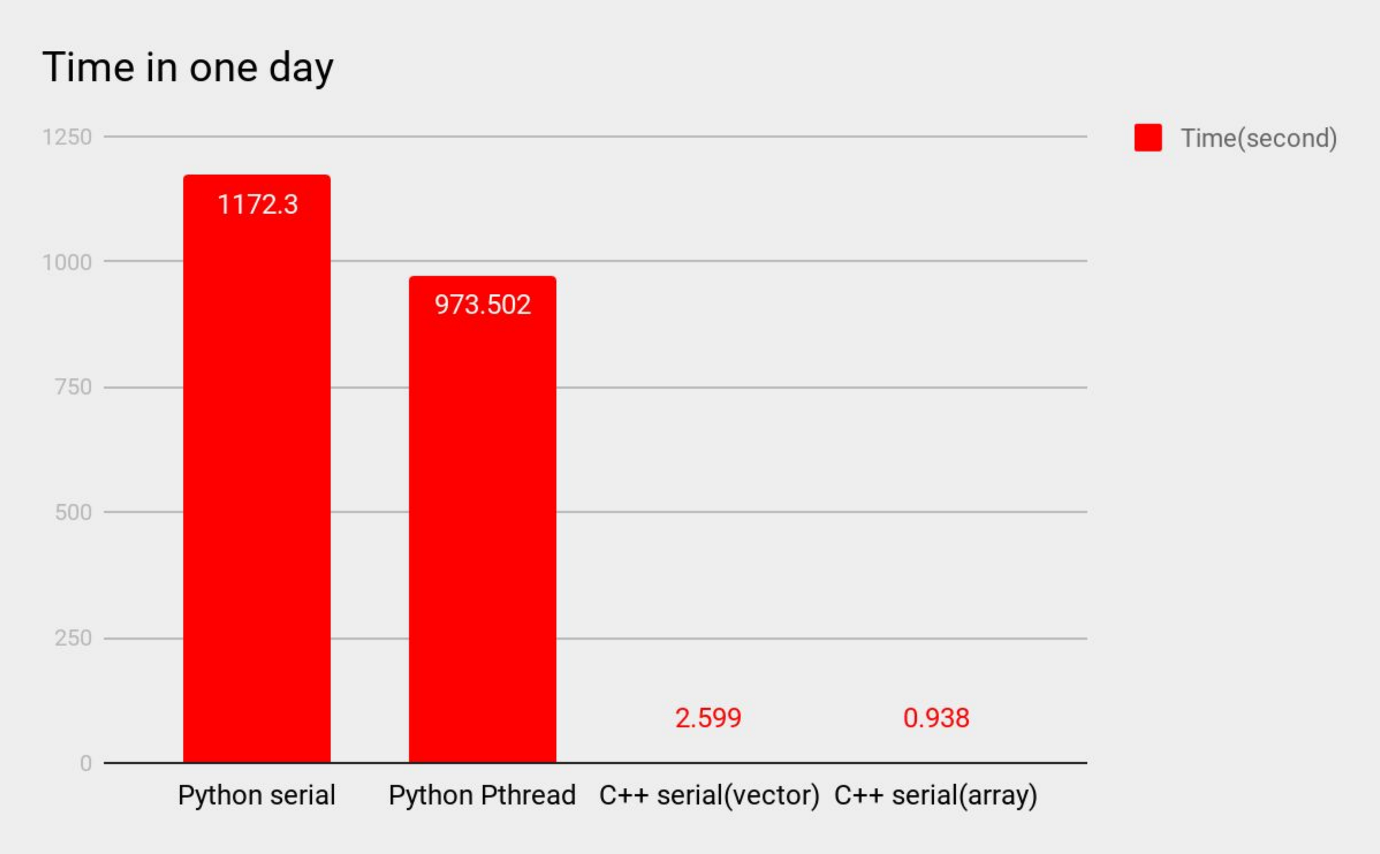
Proposed Solution

- Language and type selection :
  - Use Python to handle data preprocessing

Proposed Solution

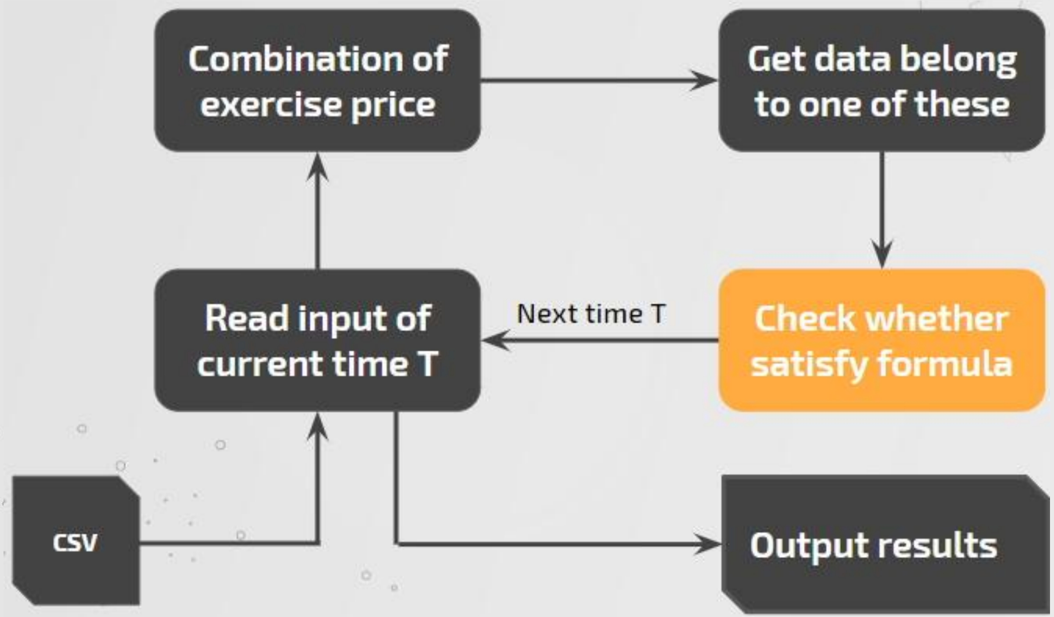
- Language and type selection :
    - Use Python to handle data preprocessing
    - Implement C++ instead, since Python needs too much time
    - We also implement vector and array version of C++

Proposed Solution



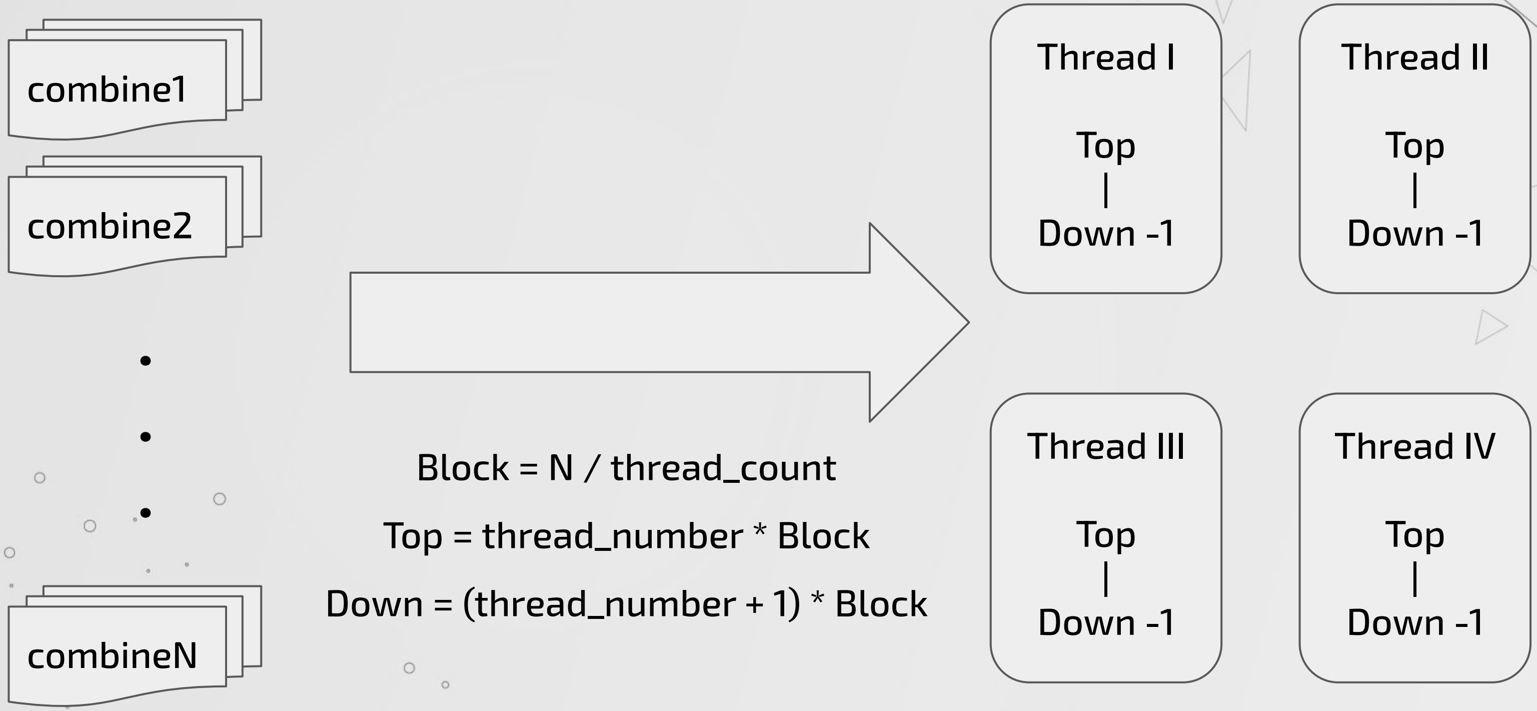
Proposed Solution

- Focus on checking all combinations satisfy formula
- Parallel model includes Pthread, OpenMP, MPI


Combination of exercise price
Get data belong to one of these
Read input of current time T
Next time T
Check whether satisfy formula
csv
Output results

Proposed Solution

- Split exercise price combination to each thread



Block = N / thread_count

Top = thread_number * Block

Down = (thread_number + 1) * Block

04

Evaluation

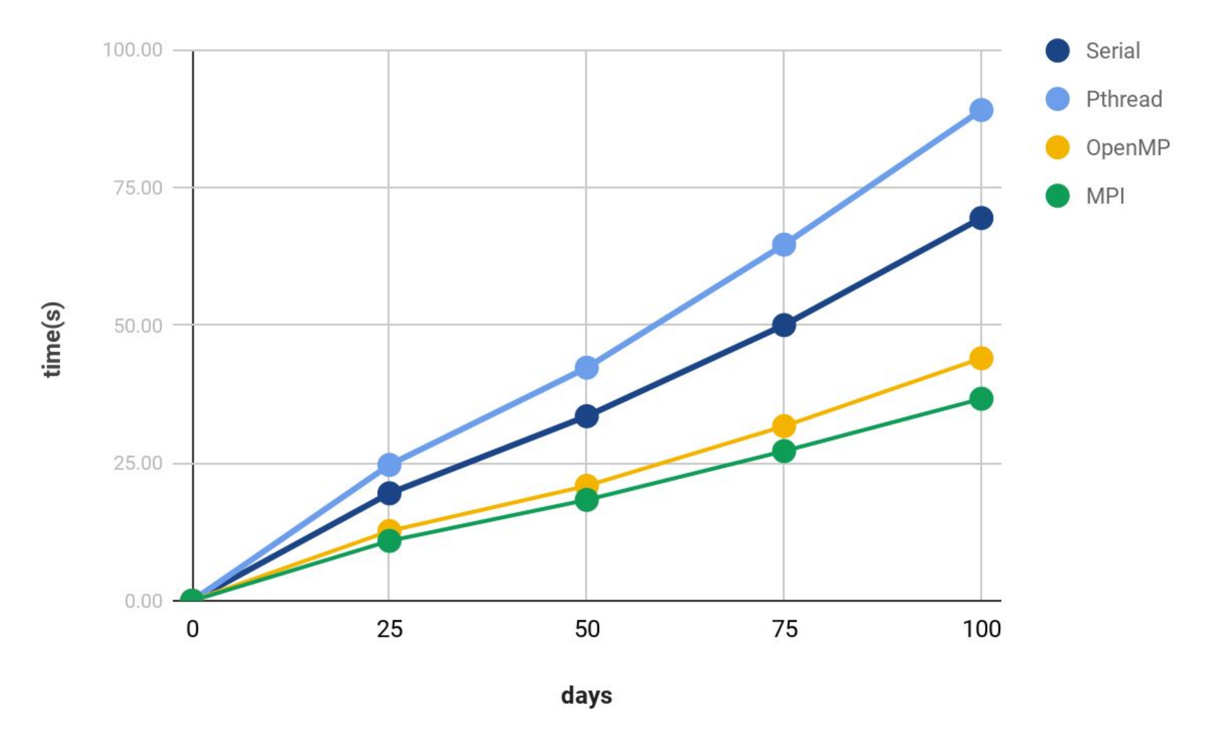
Environment

- Language : Python, C++
- Parallel model : Pthread, OpenMP, MPI
- Hardware : PP-f19 server
- Data : TAIEX Weekly Equity Index Options

  (source:https://www.taifex.com.tw/cht/3/dlOptPrevious30DaysSalesData)
- Data size : 25/50/75/100 days

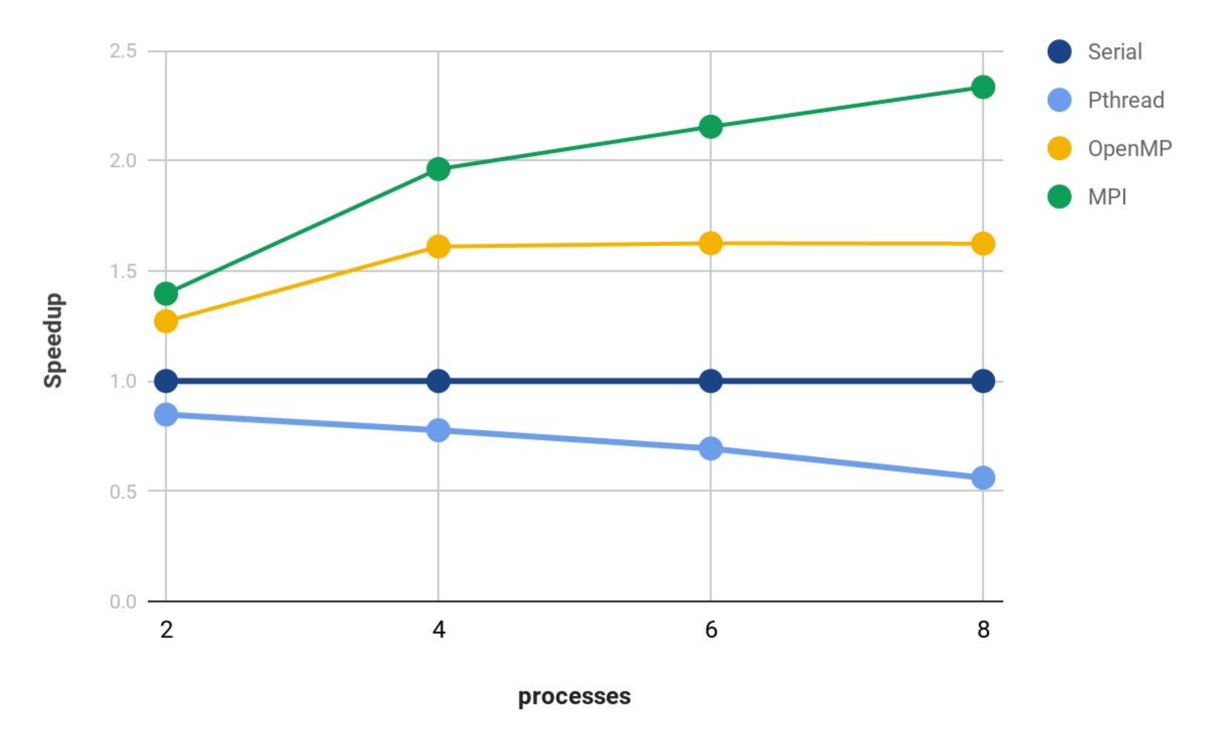
Evaluation : Time

Days	Serial	Pthread	OpenMP	MPI
25	19.49	24.65	12.66	10.87
50	33.48	42.29	20.85	18.30
75	50.01	64.61	31.68	27.15
100	69.44	89.03	43.99	36.65



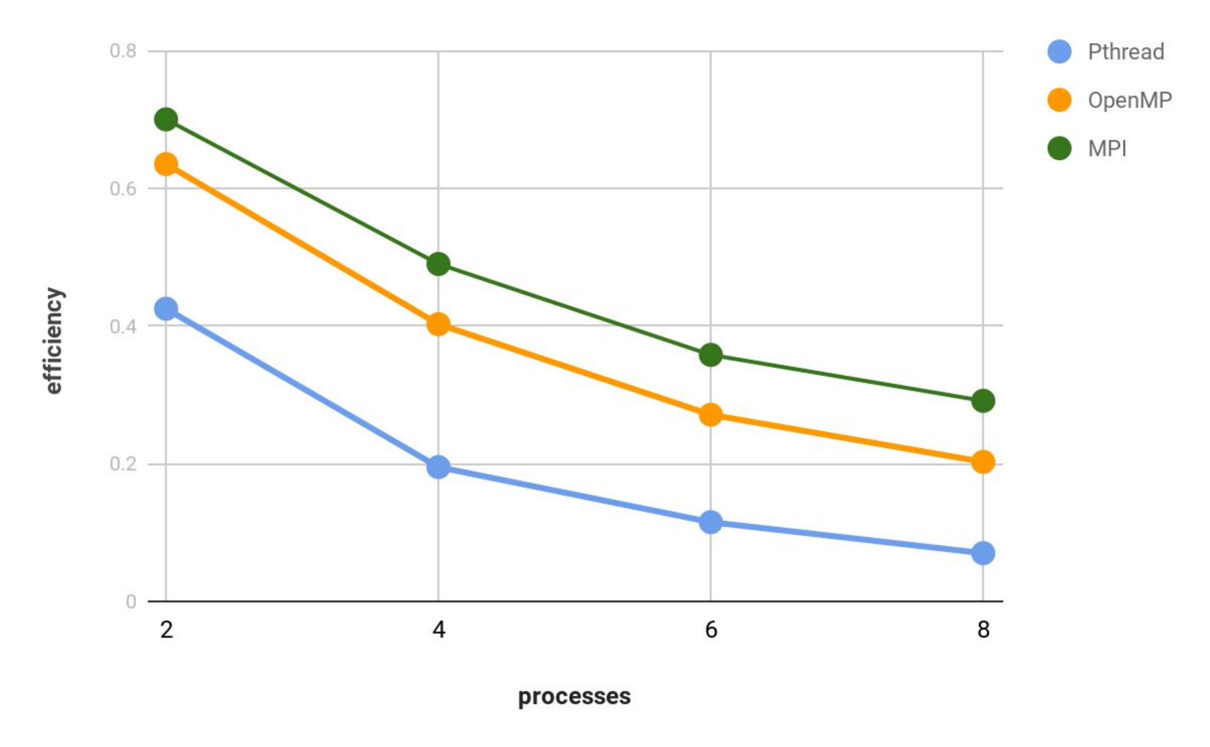
Evaluation : Speedup

process	Serial	Pthread	OpenMP	MPI
2	1	0.85	1.27	1.40
4	1	0.78	1.61	1.96
6	1	0.69	1.63	2.15
8	1	0.56	1.62	2.33



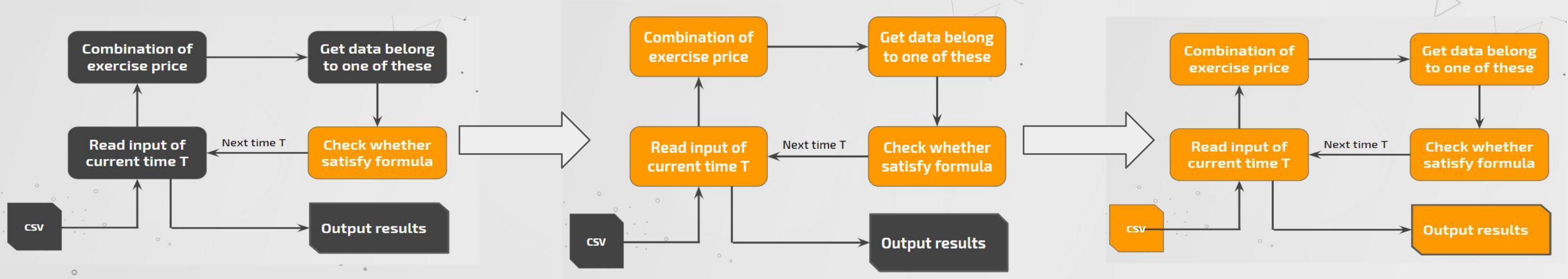
Evaluation : Efficiency

process	Serial	Pthread	OpenMP	MPI
2	1	0.425	0.635	0.7
4	1	0.195	0.403	0.49
6	1	0.115	0.271	0.358
8	1	0.07	0.203	0.29



Pthread Problem

- We assume that the reason why pthread slows down the computation is the huge overhead.
- To verify our assumption, we enlarge the calculation per thread.



Combination of exercise price

Get data belong to one of these

Read input of current time T

Next time T

Check whether satisfy formula

CSV

Output results

Combination of exercise price

Get data belong to one of these

Read input of current time T

Next time T

Check whether satisfy formula

CSV

Output results

Combination of exercise price

Get data belong to one of these

Read input of current time T

Next time T

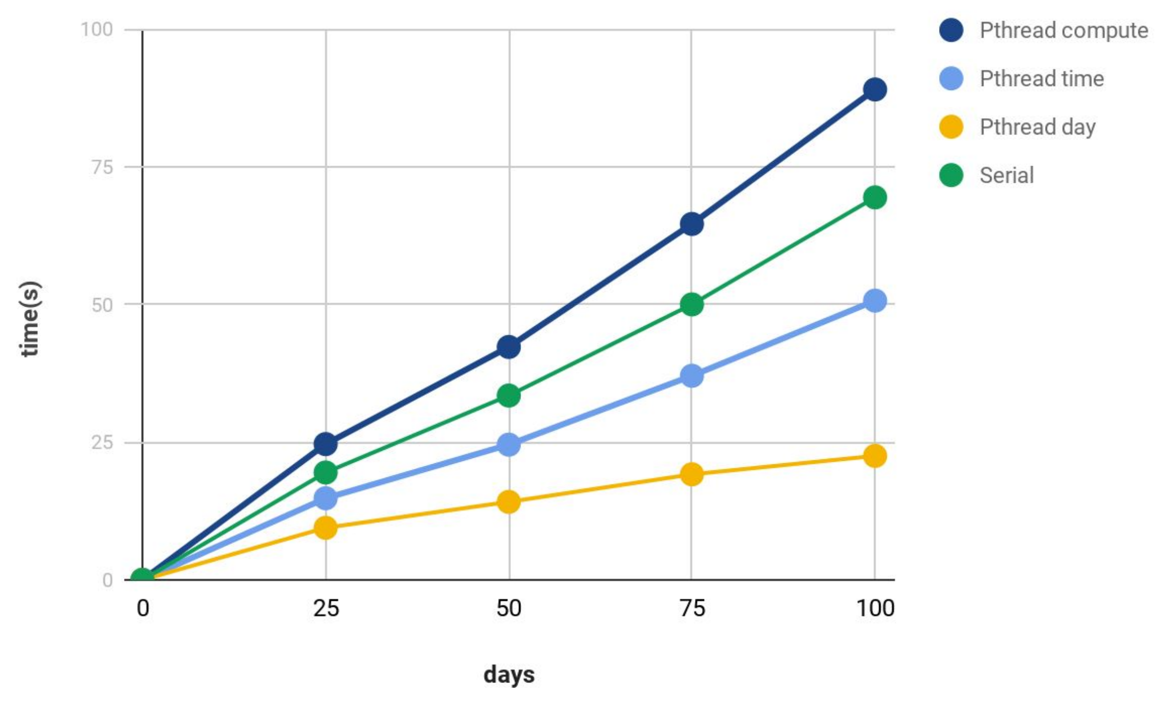
Check whether satisfy formula

CSV

Output results

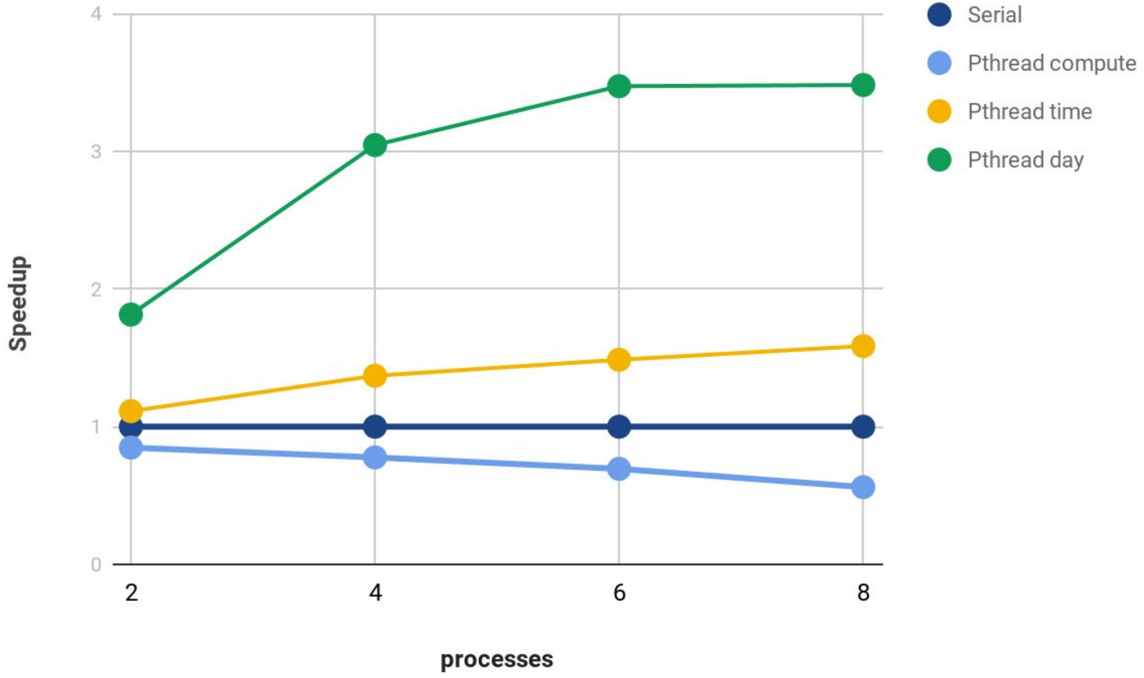
Pthread Problem

Days	Serial	Pthread compute	Pthread time	Pthread day
25	19.49	24.65	14.82	9.46
50	33.48	42.29	24.55	14.16
75	50.01	64.61	37.05	19.16
100	69.44	89.03	50.69	22.51



Pthread Problem

process	Serial	Pthread compute	Pthread time	Pthread day
2	1	0.85	1.11	1.81
4	1	0.78	1.37	3.04
6	1	0.69	1.49	3.47
8	1	0.56	1.59	3.48

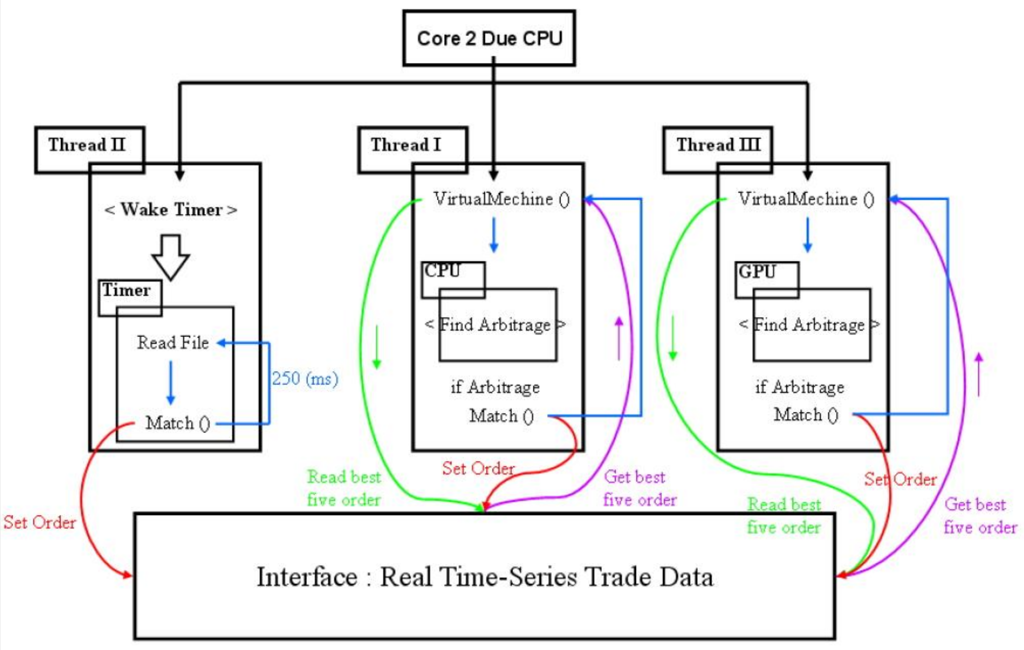


05

Related Work

Related Work

- Yu-Wen Chen,"Online Derivatives Arbitrage Trading Mechanism Based on CUDA Framework"
    - Use more sophisticated data set.
    - Parallel model : Cuda



06

Conclusion

Conclusion

- We have tried two languages and three different parallel models
- Use MPI can speedup at most 2.33 times with 8 processors than serial
- Pthread is slower than serial because of overhead
- Distributed-memory model is better than shared-memory in our case

THANKS

Q&A

Evaluation: Speedup

Mpi_array	2	49.723
	4	35.390
	6	32.234
	8	29.748

Evaluation: Speedup

openmp_array	2	54.617
	4	43.119
	6	42.719
	8	42.768
pthread _array	2	81.898
	4	89.388
	6	100.095
	8	123.781

Evaluation: Speedup

serial_array	25	19.494
	50	33.464
	75	50.010
	100	69.436
Mpi_array 4	25	10.865
	50	18.3
	75	27.152
	100	36.651
Python	1	1172.3
serial_array	1	0.938

Evaluation: Speedup

openmp_array	25	12.656
	50	20.849
	75	31.681
	100	43.990
pthread _array	25	24.650
	50	42.285
	75	64.614
	100	89.028
Python pthread	1	973.502

Evaluation: Speedup

serial_vector	25	118.143
	50	191.710
	75	287.307
	100	408.539
Mpi_vector	25	
	50	
	75	
	100	

Evaluation: Speedup

openmp_vector	25	71.123
	50	119.934
	75	183.272
	100	280.355
pthread _vecor	25	
	50	
	75	
	100	

Evaluation: Speedup

Divide intratime 4	25	14.821
	50	24.549
	75	37.054
	100	50.694
Divide day 4	25	9.455
	50	14.163
	75	19.163
	100	22.511

Evaluation: Speedup

Divide intratime(days=100)	2	62.396
	4	50.694
	6	46.730
	8	43.802
Divide day(days=100)	2	38.276
	4	22.806
	6	19.996
	8	19.948