



ΔΗΜΟΚΡΙΤΟΣ
ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΕΡΕΥΝΑΣ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS



Report εργασίας Automata

Αλεξίου Κυριάκος ΑΜ: 2303

Γεωργούλας Σπυρίδων ΑΜ: 2309

Ρωμέσης Χριστόφορος ΑΜ: 2318

24 Ιουλίου, 2024

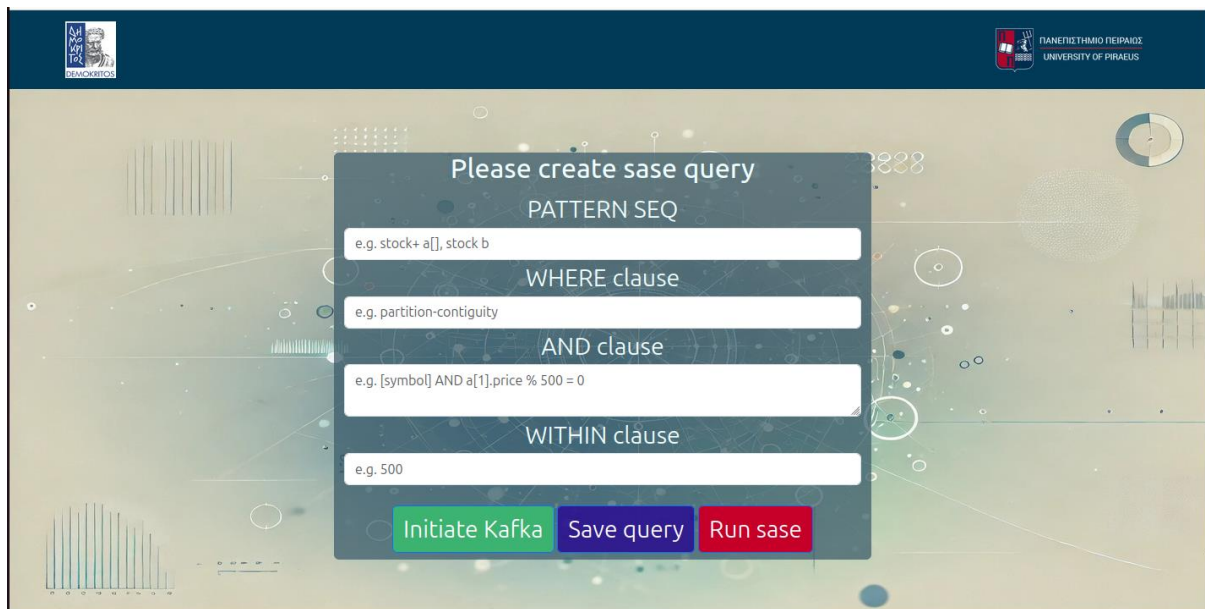
Η παρούσα εργασία σκοπό έχει να δημιουργήσει ένα framework με το οποίο θα μπορούν να γίνονται πειράματα με διαφορετικά query πάνω σε stream το οποίο τροφοδοτείται μέσω kafka (distributed event streaming platform). Σκοπός της εργασίας είναι να “παρέχει” όλα τα απαραίτητα εργαλεία με τα οποία ο χρήστης θα μπορεί να πειραματιστεί σε διάφορα queries σε εισερχόμενο stream

Χρησιμοποιείται μια web εφαρμογή γραμμένη σε python (Django) υπεύθυνη για την διεπαφή με τον χρήστη, ο apache kafka (dockerized) καθώς και μια τροποποιημένη έκδοση του sase.jar αρχείου το οποίο δύναται να διαβάσει messages από τον kafka σε πραγματική ροή και με βάση το query που έχουμε εισάγει στην web εφαρμογή να παρουσιάσει τα αποτελέσματα. Όλα τα dependencies καθώς και βοηθητικά προγράμματα που θα αναλυθούν παρακάτω, βρίσκονται αναρτημένα στο παρακάτω repo:

https://github.com/kiriakos2004/sase_kafka.git

Web app

Η εφαρμογή όπως προαναφέρθηκε είναι γραμμένη σε python με χρήση του Django full stack. Στη εικόνα 1 φαίνεται η αρχική σελίδα:

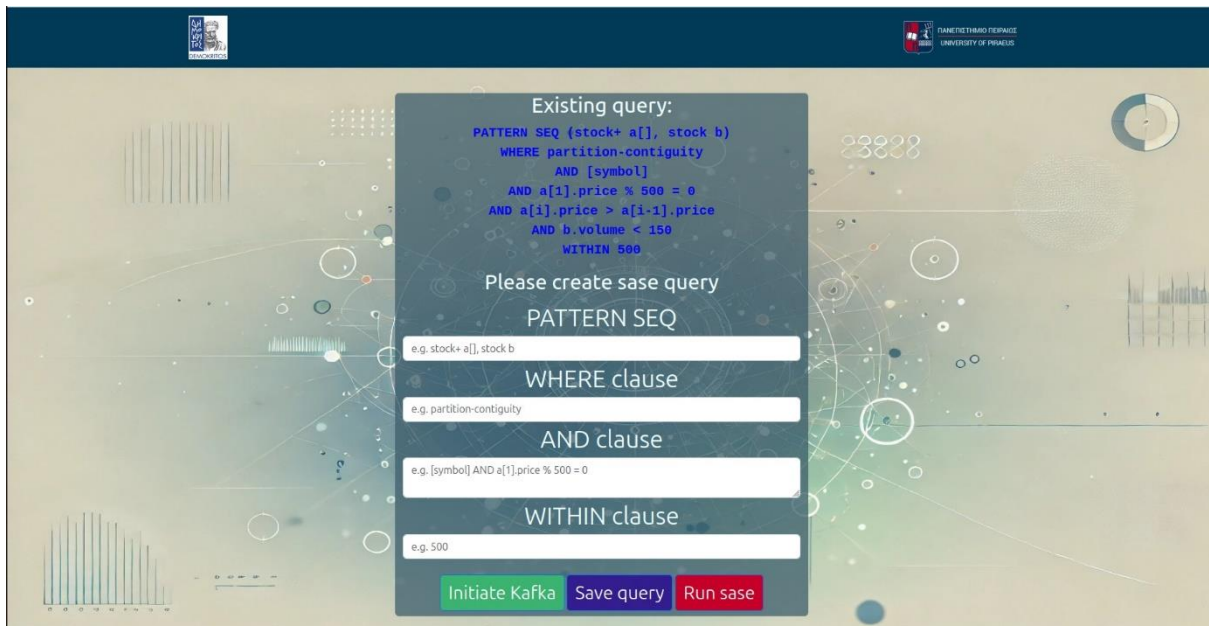
The image shows a web application interface with a dark blue header. On the left is a small logo, and on the right is the text 'ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ UNIVERSITY OF PIRAEUS'. The main content area has a light beige background with a faint, abstract pattern. A central dark grey form titled 'Please create sase query' contains four input fields: 'PATTERN SEQ' (with example 'e.g. stock+ a[], stock b'), 'WHERE clause' (with example 'e.g. partition-contiguity'), 'AND clause' (with example 'e.g. [symbol] AND a[1].price % 500 = 0'), and 'WITHIN clause' (with example 'e.g. 500'). At the bottom of the form are three buttons: 'Initiate Kafka' (green), 'Save query' (blue), and 'Run sase' (red).

Εικ.1

Στην σελίδα αυτή ο χρήστης μπορεί να ενεργοποιήσει τον kafka (λεπτομέρειες σε επόμενο κεφάλαιο) καθώς και να τροποποιήσει κατά το δοκούν το query που θέλει να εφαρμόσει στο stream. Ο χρήστης πατώντας το κουμπί “save query” τροποποιεί το αρχείο test.query που χρησιμοποιείται από το sase.jar πρόγραμμα ως οδηγίες για το query. Ο χρήστης μπορεί να ορίσει την διαδρομή αυτού του αρχείου εδώ:

```
8
9 def home(request):
10     file_content = None
11
12     if request.method == 'POST':
13         pattern = request.POST.get('pattern', '').strip()
14         where_clause = request.POST.get('where', '').strip()
15         and_clause = request.POST.get('and', '').strip()
16         within_clause = request.POST.get('within', '').strip()
17
18         if pattern and where_clause and and_clause and within_clause:
19             file_content = f"PATTERN SEQ ({pattern})\nWHERE {where_clause}\nAND {and_clause}\nWITHIN {within_clause}"
20
21         # Define the directory and file path
22         directory_path = '/home/kiriakos/Documents/SASE HOME MSC/build'
23         file_path = os.path.join(directory_path, 'test.query')
24
25         # Ensure the directory exists
26         if not os.path.exists(directory_path):
27             os.makedirs(directory_path)
28
29         # Write the content to the file
30         with open(file_path, 'w') as file:
31             file.write(file_content)
32
33         # Read the content from the file
34         with open(file_path, 'r') as file:
35             file_content = file.read()
36
37     return render(request, 'home.html', {'file_content': file_content})
```

Το εν ισχύ query εμφανίζεται στον χρήστη στην σελίδα:



Με χρήση του κουμπιού “Run sase” εκτελείται η εντολή “java – jar sase. jar test.query” (στον φάκελο στον οποίο υπάρχει το sase.jar καθώς και το ανωτέρω test.query που δημιουργήσαμε). Ο φάκελος αυτός αλλάζει εδώ :

```
DEMOCRITOS
└─ klimakosimes
   └─ klimakosimes
      └─ sase
         ├── _pycache_
         ├── migrations
         ├── templates
         ├── __init__.py
         ├── admin.py
         ├── apps.py
         ├── models.py
         ├── tests.py
         ├── urls.py
         └── views.py
            ├── static
            ├── templates
            ├── db.sqlite3
            ├── manage.py
            ├── lib
            ├── lib64
            ├── 227592820NR.csv
            ├── custom.cnf
            ├── docker-compose.yml
            ├── kafka_consumer_db.py
            ├── kafka_consumer.py
            ├── kafka_producer.py
            ├── LICENSE.txt
            └── main.py

klimakosimes > sase > views.py > initiate_sase
37     return render(request, 'home.html', {'file_content': file_content})
38
39     @csrf_exempt
40     @require_POST
41     def initiate_sase(request):
42         try:
43             # Absolute path to the directory containing sase.jar, test.query, and test.stream
44             sase_dir = '/home/kiriakos/Documents/SASE HOME MSC/build'
45             jar_path = 'sase.jar'
46             result = subprocess.run(
47                 ['java', '-jar', jar_path, test.query],
48                 cwd=sase_dir,
49                 capture_output=True,
50                 text=True
51             )
52             request.session['sase_output'] = result.stdout
53             request.session['sase_error'] = result.stderr
54             return JsonResponse({'status': 'success', 'redirect_url': reverse('sase_result')})
55         except Exception as e:
56             return JsonResponse({'status': 'error', 'error': str(e)})
57
58     def sase_result(request):
59         output = request.session.get('sase_output', 'No output available')
60         error = request.session.get('sase_error', '')
61         return render(request, 'sase_result.html', {'output': output, 'error': error})
62
63     @csrf_exempt
64     @require_POST
65     def run_docker_compose(request):
66         try:
```

Με το ανωτέρω query και με χρήση του “Run sase” επιστρέφονται τα αποτελέσματα:



Sase Results:

Repeat No.1 is started...

-----Here is the No.1 match-----

This match has selected the following events:

ID = 723	Timestamp = 723	Symbol = 1	Price = 588	Volume = 386
ID = 724	Timestamp = 724	Symbol = 1	Price = 562	Volume = 64

-----Here is the No.2 match-----

This match has selected the following events:

ID = 1164	Timestamp = 1164	Symbol = 2	Price = 1800	Volume = 380
ID = 1165	Timestamp = 1165	Symbol = 2	Price = 1800	Volume = 57

-----Here is the No.3 match-----

This match has selected the following events:

ID = 1177	Timestamp = 1177	Symbol = 2	Price = 1800	Volume = 815
ID = 1179	Timestamp = 1179	Symbol = 2	Price = 1801	Volume = 484
ID = 1183	Timestamp = 1183	Symbol = 2	Price = 1802	Volume = 798
ID = 1184	Timestamp = 1184	Symbol = 2	Price = 1804	Volume = 23

-----Here is the No.4 match-----

This match has selected the following events:



ID = 4247	Timestamp = 4247	Symbol = 2	Price = 2580	Volume = 527
ID = 4248	Timestamp = 4248	Symbol = 2	Price = 2582	Volume = 384
ID = 4251	Timestamp = 4251	Symbol = 2	Price = 2595	Volume = 84

-----Here is the No.5 match-----

This match has selected the following events:

ID = 5847	Timestamp = 5847	Symbol = 2	Price = 3888	Volume = 539
-----------	------------------	------------	--------------	--------------

Ένα δεύτερο παράδειγμα:

Existing query:

```
PATTERN SEQ (stock+ a[], stock b)
WHERE partition-contiguity
AND [symbol]
AND a[i].price > a[i-1].price
AND b.volume > 200
WITHIN 800
```

Please create sase query

PATTERN SEQ

e.g. stock+ a[], stock b

WHERE clause

e.g. partition-contiguity

AND clause



e.g. [symbol] AND a[i].price % 500 = 0

WITHIN clause

e.g. 500

Initiate Kafka
Save query
Run sase

Με τα αποτελέσματα:

Sase Results:

```
Repeat No.1 is started...
-----Here is the No.1 match-----

This match has selected the following events:

ID = 0 Timestamp = 0 Symbol = 1 Price = 111 Volume = 457
ID = 1 Timestamp = 1 Symbol = 1 Price = 110 Volume = 215

-----Here is the No.2 match-----

This match has selected the following events:

ID = 1 Timestamp = 1 Symbol = 1 Price = 110 Volume = 215
ID = 2 Timestamp = 2 Symbol = 1 Price = 109 Volume = 209

-----Here is the No.3 match-----

This match has selected the following events:

ID = 2 Timestamp = 2 Symbol = 1 Price = 109 Volume = 209
ID = 4 Timestamp = 4 Symbol = 1 Price = 109 Volume = 239

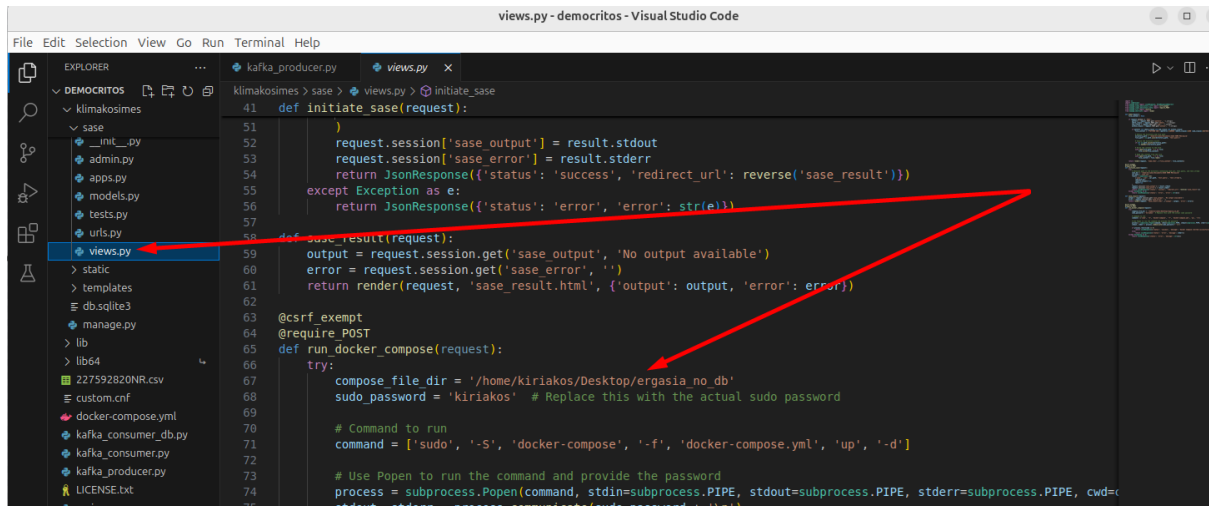
-----Here is the No.4 match-----

This match has selected the following events:

ID = 4 Timestamp = 4 Symbol = 1 Price = 109 Volume = 239
ID = 5 Timestamp = 5 Symbol = 1 Price = 110 Volume = 509
```

Kafka server

Στο πλαίσιο της υλοποίησης της εφαρμογής με χρήση kafka, δημιουργήθηκε ένα docker-compose.yml file το οποίο σηκώνει τα τρία απαραίτητα containers που απαιτούνται για την ενεργοποίηση και χρήση του kafka (kafka, kafka-manager και zookeeper). Η ενεργοποίηση των containers μπορεί να γίνει μέσω του button “initiate kafka” από το web UI. Το web UI καλεί την εντολή `sudo docker compose up -d` από συγκεκριμένο φάκελο στον οποίο είναι αποθηκευμένο το yml. Η διαδρομή αυτή καθορίζεται εδώ:



Kafka producer / consumer

Για την υλοποίηση των πειραμάτων με χρήση του kafka δημιουργήθηκε κώδικας με χρήση python (kafka_producer.py) ο οποίος διαβάζει ένα csv file (stream_data_200000.csv) και κάνει post στο topic test μια γραμμή του csv κάθε 1/10 του δευτερολέπτου.

Ο τροποποιημένος κώδικας java ο οποίος κάνει subscribe στο ανωτέρω topic και κάνει consume τα messages βρίσκεται στον φάκελο build_kafka σε compiled μορφή .jar (sase.jar) καθώς και οι φάκελοι με τα readable .java αρχεία από τα οποία αποτελείται. (Τροποποιήθηκαν τα αρχεία CommandLineUI.java καθώς και StreamController.java).

Στον φάκελο build kafka υπάρχει το αρχείο Kafka_results.log στο οποίο φαίνονται τα αποτελέσματα της επεξεργασίας του ανωτέρω stream που μέσω kafka επεξεργάστηκε το νέο πρόγραμμα sase.jar που δημιουργήθηκε. Για την πραγματοποίηση του πειράματος χρησιμοποιήθηκε το query:

```
PATTERN SEQ(stock+ a[], stock b)
WHERE partition-contiguity
AND [symbol]
AND a[i].price > a[i-1].price
AND b.volume < 200
WITHIN 900
```

το οποίο επίσης βρίσκεται στον ίδιο φάκελο.

Στην παρακάτω εικόνα φαίνεται ενδεικτικά το throughput του sase κατά την χρήση του σε συνεργασία με το kafka:

Profiling results for repeat No.2 are as follows:

*****Profiling Numbers*****

Total Running Time: 52239807 nanoseconds

Number Of Events Processed: 1000

Number Of Runs Created: 1000

Number Of Matches Found: 286

Throughput: 19142 events/second

- . - . - .

Για περισσότερες τιμές throughput μπορούμε να ανατρέξουμε στο αρχείο Kafka_results.log

Παραδείγματα εφαρμογών

Στην συνέχεια χρησιμοποιούμε το SASE για τον εντοπισμό 3 περιπτώσεων complex events από το χώρο των χρηματοπιστωτικών συναλλαγών μετοχών.

ΠΑΡΑΔΕΙΓΜΑ-1: ΕΝΤΟΠΙΣΜΟΣ ΞΑΦΝΙΚΗΣ ΡΑΓΔΑΙΑΣ ΠΤΩΣΗΣ ΤΙΜΗΣ

Ζητείται να εντοπιστούν περιπτώσεις μετοχών με υψηλή τιμή και κίνηση, που ενώ ο όγκος των συναλλαγών τους δεν σημειώνει σημαντική μεταβολή, η τιμή της σημειώνει ξαφνικά ραγδαία πτώση (άνω του 30%).

test.query

```
PATTERN SEQ(stock+ a[], stock b)
WHERE partition-contiguity
AND [symbol]
AND a[1].price > 2000
AND a[1].volume > 4000
AND a[i].volume > 95%*a[i-1].volume
AND a[i].volume < 105%*a[i-1].volume
AND b.price < 70%*a[1].price
WITHIN 500
```

SASE RESULTS

-----Here is the No.8805 match-----

This match has selected the following events:

ID = 99962	Timestamp = 99962	Symbol = 1	Price = 54648	Volume = 6040
ID = 99963	Timestamp = 99963	Symbol = 1	Price = 54645	Volume = 286

-----Here is the No.8806 match-----

This match has selected the following events:

ID = 99967	Timestamp = 99967	Symbol = 2	Price = 55723	Volume = 4455
ID = 99968	Timestamp = 99968	Symbol = 2	Price = 55720	Volume = 6153

-----Here is the No.8807 match-----

This match has selected the following events:

ID = 99966	Timestamp = 99966	Symbol = 1	Price = 54650	Volume = 7767
ID = 99969	Timestamp = 99969	Symbol = 1	Price = 54647	Volume = 5928

Profiling results for repeat No.20 are as follows:

*****Profiling Numbers*****

Total Running Time: 774125475 nanoseconds

Number Of Events Processed: 100000

Number Of Runs Created: 58018

Number Of Matches Found: 8807

Throughput: 129178 events/second

ΠΑΡΑΔΕΙΓΜΑ-2: ΕΝΤΟΠΙΣΜΟΣ ΚΑΤΑΛΛΗΛΗΣ ΣΤΙΓΜΗΣ ΑΓΟΡΑΣ

Ζητείται να εντοπιστούν περιπτώσεις μετοχών με υψηλή αρχική τιμή και κίνηση, που αρχίζει να πέφτει η τιμή της και εντοπίζουμε τη στιγμή που η τιμή αυτή αρχίζει και πάλι να ανεβαίνει. Το σημείο που η τιμή θα ξεπεράσει το φράγμα του 75% της αρχικής τιμής, σηματοδοτεί το σημείο που θεωρείται κατάλληλο να αγοράσουμε.

test.query

PATTERN SEQ(stock+ a[], stock+ b[], stock c)

WHERE partition-contiguity

AND [symbol]

AND a[1].price > 3000

AND a[1].volume > 4000

AND a[i].price < a[i-1].price

AND a[a.LEN].price < 70%*a[1].price

AND b[i].price > b[i-1].price

AND c.price > 75%*a[1].price

WITHIN 1000

SASE RESULTS

-----Here is the No.40023 match-----

This match has selected the following events:

ID = 99992	Timestamp = 99992	Symbol = 2	Price = 55732	Volume = 4750
ID = 99993	Timestamp = 99993	Symbol = 2	Price = 55735	Volume = 7371

ID = 99995 Timestamp = 99995 Symbol = 2 Price = 55736 Volume = 2507

-----Here is the No.40024 match-----

This match has selected the following events:

ID = 99993	Timestamp = 99993	Symbol = 2	Price = 55735	Volume = 7371
ID = 99995	Timestamp = 99995	Symbol = 2	Price = 55736	Volume = 2507
ID = 99996	Timestamp = 99996	Symbol = 2	Price = 55739	Volume = 6279

Profiling results for repeat No.20 are as follows:

*****Profiling Numbers*****

Total Running Time: 3872803715 nanoseconds

Number Of Events Processed: 100000

Number Of Runs Created: 56958

Number Of Matches Found: 40024

Throughput: 25821 events/second

ΠΑΡΑΔΕΙΓΜΑ-2: ΕΝΤΟΠΙΣΜΟΣ ΤΑΒΑΝΙΟΥ ΤΙΜΗΣ ΜΕΤΟΧΗΣ

Ζητείται να εντοπιστούν περιπτώσεις μετοχών που αρχικά η τιμή τους ανεβαίνει, μετά ακολουθεί μια περίοδος με σχεδόν αμετάβλητη τιμή και τέλος ακολουθεί μια πτωτική πορεία της τιμής μέχρι την τιμή 2000.

test.query

PATTERN SEQ(stock+ a[], stock+ b[], stock+ c[], stock d)

WHERE partition-contiguity

AND [symbol]

AND a[i].price > a[i-1].price

AND b[i].price > 95%*b[i-1].price

AND b[i].price < 105%*b[i-1].price

AND c[i].price < c[i-1].price

AND d.price < 2000

WITHIN 1000

SASE RESULTS

-----Here is the No.3389 match-----

This match has selected the following events:

ID = 3415	Timestamp = 3415	Symbol = 1	Price = 1999	Volume = 6293
ID = 3416	Timestamp = 3416	Symbol = 1	Price = 1996	Volume = 5429
ID = 3419	Timestamp = 3419	Symbol = 1	Price = 1998	Volume = 9114
ID = 3422	Timestamp = 3422	Symbol = 1	Price = 1998	Volume = 1386

-----Here is the No.3390 match-----

This match has selected the following events:

ID = 3422	Timestamp = 3422	Symbol = 1	Price = 1998	Volume = 1386
ID = 3423	Timestamp = 3423	Symbol = 1	Price = 2001	Volume = 5692
ID = 3425	Timestamp = 3425	Symbol = 1	Price = 2001	Volume = 7308
ID = 3426	Timestamp = 3426	Symbol = 1	Price = 1999	Volume = 1876

Profiling results for repeat No.20 are as follows:

*****Profiling Numbers*****

Total Running Time: 671422979 nanoseconds

Number Of Events Processed: 100000

Number Of Runs Created: 100000

Number Of Matches Found: 3390

Throughput: 148937 events/second
