

Przykładowy sprawdzian wykładowy

Sprawdzian składa się z kilku pytań o różnej skali trudności. Część z nich wymaga znajomości teorii omawianej na wykładzie, część znajomości poznanych technologii i silników przetwarzania strumieni danych, a część opartych o zdobytą wiedzę obliczeń i symulacji.

Krótką charakterystyka pytań:

1. podstawy znaczników watermark, wyzwalaczy, obsługi i typów okien
2. znajomość poznanych produktów i technologii w kontekście mechanizmów związanych ze znacznikami watermark, wyzwalaczami, obsługą i typami okien
3. input + output -> charakterystyka przetwarzania – na podstawie podanych danych wejściowych oraz wyniku przetwarzania, należy określić charakterystykę przetwarzania, w tym: typ wyzwalacza, tryb obsługi znaczników watermark, tryb obsługi okien itp.
4. charakterystyka przetwarzania + input -> output – na podstawie podanych danych wejściowych oraz charakterystyki przetwarzania, w tym: typu wyzwalacza, tryby obsługi znaczników watermark, tryby obsługi okien itp. należy określić wynik przetwarzania
5. punkt kontrolny wyrównujący – dla podanych danych wejściowych oraz wskazanego momentu uruchomienia punktu kontrolnego należy określić jego zawartość
6. punkt kontrolny niewyrównujący – dla podanych danych wejściowych oraz wskazanego momentu uruchomienia punktu kontrolnego należy określić jego zawartość

Każde zadanie jest tak samo punktowane. Nie oznacza to jednak, że te nie wymagające obliczeń i symulacji są najprostsze. Na takich najszybciej zyskuje się punkty, ale też najszybciej się traci. Podczas sprawdzianu skoncentrować się na tych tematach, na których się znamy i których jesteśmy pewni. Dopiero potem, kiedy czas na to pozwoli idziemy w jakość i staramy się poprawić ostateczny wynik.

Poniżej znajdują się przykładowe zadania z każdej kategorii

1. Podstawy znaczników watermark, wyzwalaczy, obsługi i typów okien

Jakiego typu wyzwalacze generują największe ilości danych na wyjściu (do ujęcia)?

(zaznacz wiele odpowiedzi jeśli wiele typów wyzwalaczy jest "identycznych" pod względem powyższego kryterium)

- ☐ a. Wyzwalacz z wyrównanym opóźnieniem
- ☐ b. Wyzwalacz kompletności
- ☐ c. Wyzwalacz z niewyrównanym opóźnieniem
- ☐ d. Natychmiastowy wyzwalacz powtarzalnej aktualizacji

2. Znajomość poznanych produktów i technologii

W systemie przetwarzania strumieni danych Apache Flink programista zdefiniował następujące klasy

```
public class InputScores {  
    private String house;  
    private String character;  
    private String score;  
    private String ts;  
    . . .  
  
    public class HouseStats {  
        private String house;  
        private long sumScore;  
        . . .
```

każda z klas ma odpowiednie konstruktory, gettery oraz settery.

Ponadto programista zdefiniował funkcję, która dostarcza źródło KafkaSource

```
public static KafkaSource<String> getKafkaSource(ParameterTool properties) {  
    . . .
```

a następnie zapisał następujący kod:

```
KafkaSource<String> source = Connectors.getKafkaSource(properties);  
  
DataStream<String> inputDS = env.fromSource(source, WatermarkStrategy.noWatermarks(), "Kafka Source");  
  
// ustrukturalizowanie danych  
DataStream<InputScores> inputScoresDS = inputDS.flatMap(new InputScoresFlatMap());  
  
DataStream<HouseStats> houseStatsDS = inputScoresDS  
    .map(is -> new HouseStats(is.getHouse(), Integer.parseInt(is.getScore())));  
  
DataStream<HouseStats> resultDS = houseStatsDS  
    .keyBy(HouseStats::getHouse)  
    .window(TumblingProcessingTimeWindows.of(Time.seconds(10)))  
    .reduce((v1, v2) -> new HouseStats(v1.getHouse(), v1.getSumScore() + v2.getSumScore()));  
  
resultDS.print();  
  
env.execute();
```

Przeanalizuj powyższy kod, a następnie określ czy jest on poprawny z punktu widzenia zastosowanych transformacji oraz trybu obsługi wyniku, oraz jeśli tak, to jaki typ wyzwalacza został tu zastosowany.

- ☐ Program jest poprawny. Wyzwalacz wcześnie/na czas/późno (EOTL)
- ☐ Program jest poprawny. Wyzwalacz z wyrównanym opóźnieniem
- ☐ Program jest poprawny. Wyzwalacz kompletności
- ☐ Program jest niepoprawny. Tryb wyniku jest niekompatybilny z zastosowanymi transformacjami
- ☐ Program jest poprawny. Wyzwalacz z niewyrównanym opóźnieniem
- ☐ Program jest poprawny. Natychmiastowy wyzwalacz powtarzalnej aktualizacji

3. Input + output -> charakterystyka przetwarzania

Programista działa na strumieniu zdarzeń

```
DataStream<SensorData> inputSource = env.addSource(...);
```

gdzie klasa SensorData zdefiniowana jest następująco

```
public class SensorData {  
    private Integer value;  
    private String sensor;  
    ... wymagane konstruktory, gettery i settery  
}
```

Zdarzenia w strumieniu mają przypisane na wejściu etykiety czasowe zdarzeń.

i na tych tylko etykietach opiera się system przetwarzania strumieni danych.

Program, który został napisany przez programistę wylicza dla każdego sensora sumy wartości.

Niektóre szczegóły implementacyjne

- okno *tumbling* o długości 4 (okna mają domknięte granice początkowe i otwarte granice końcowe)

Poniżej znajdują się dane wejściowe oznaczone jako

```
I;etkieta_czasowa_zdarzenia> [sensor, value]
```

oraz dane wynikowe oznaczone jako

```
O;etkieta_czasowa_zdarzenia_wynikowego> [sensor, value]
```

```
I;00> [A, 04]  
O;03> [A, 04]  
I;01> [A, 02]  
O;03> [A, 06]  
I;06> [A, 08]  
O;07> [A, 08]  
O;03> [A, 06]  
I;02> [B, 07]  
I;05> [B, 01]  
O;07> [B, 01]  
I;03> [B, 09]  
I;07> [B, 01]  
O;07> [B, 02]  
I;09> [A, 06]  
O;11> [A, 06]  
O;07> [A, 08]  
O;07> [B, 02]  
I;04> [B, 09]  
I;08> [A, 03]  
O;11> [A, 09]  
I;11> [B, 06]  
O;11> [B, 06]  
I;10> [B, 05]  
O;11> [B, 11]  
O;11> [A, 09]  
O;11> [B, 11]
```

Założ, że na końcu tego strumienia wysyłany jest znacznik watermark o wartości +nieskończoność

Dla powyższych danych określ pozostałe szczegóły implementacyjne

Typ wyzwalacza:

- ☐ Natychmiastowy wyzwalacz powtarzalnej aktualizacji
- ☐ Wyzwalacz kompletności
- ☐ Wyzwalacz wcześniej (natychmiastowy) /na czas (EOT)
- ☐ Wyzwalacz na czas/późno (natychmiastowy) (EOTL)

Dopuszczalne opóźnienie

- ☐ Dopuszczalne opóźnienie wynosi 3
- ☐ Wynosi 0 (watermark jest tożsamy z etykietami czasowymi zdarzeń)

Zawartość kolejnych tafli w ramach tego samego okna oparta

- ☐ na podejściu akumulacyjnym z wycofaniem
- ☐ na podejściu akumulacyjnym
- ☐ na podejściu z porzucaniem
- ☐ nie dotyczy - generowana jest tylko jedna tafla dla danego okna

4. Charakterystyka przetwarzania + input -> output

Programista działa na strumieniu zdarzeń

```
DataStream<SensorData> inputSource = env.addSource(...);
```

gdzie klasa SensorData zdefiniowana jest następująco

```
public class SensorData {  
    private Integer value;  
    private String sensor;  
    ... wymagane konstruktory, getters i setters  
}
```

Zdarzenia w strumieniu mają przypisane na wejściu etykiety czasowe zdarzeń.

i na tych tylko etykietach opiera się system przetwarzania strumieni danych.

Program, który został napisany przez programistę wylicza dla każdego sensora sumy wartości.

Szczegóły implementacyjne:

- okno *tumbling* o długości 4 (okna mają domknięte granice początkowe i otwarte granice końcowe)
- wyzwalacz na czas/późno (natychmiastowy) (OTL)
- zawartość kolejnych tafli w ramach tego samego okna oparta na podejściu akumulacyjnym
- dopuszczalne opóźnienie wynosi 0 (watermark jest tożsamy z etykietami czasowymi zdarzeń)
- obsługa zdarzeń spóźnionych przez długość czasu 3

Dane wejściowe mają następujący format:

```
I;etykieta_czasowa_zdarzenia> [sensor, value]
```

i są następujące:

```
I;00> [B, 02]  
I;02> [A, 08]  
I;04> [B, 04]  
I;03> [B, 09]  
I;01> [A, 08]  
I;07> [B, 05]  
I;10> [A, 03]  
I;05> [A, 03]  
I;09> [A, 03]  
I;08> [B, 06]  
I;11> [A, 09]  
I;06> [A, 05]
```

Założ, że na końcu tego strumienia wysyłany jest znacznik watermark o wartości +nieskończoność

Dla powyższych danych wejściowych określ dane wysyłane do ujścia.

Podaj je w następujący sposób

```
etykieta_czasowa_zdarzenia_wyzwalajacego_generowanie_tafli;etykieta_czasowa_zdarzenia_wynikowego [sensor, value]
```

Ogranicz się tylko do początkowych zdarzeń umieszczanych w strumieniu wynikowym.

<input type="text"/>	;	<input type="text"/>	>	[B,	<input type="text"/>]
<input type="text"/>	;	<input type="text"/>	>	[A,	<input type="text"/>]
<input type="text"/>	;	<input type="text"/>	>	[B,	<input type="text"/>]
<input type="text"/>	;	<input type="text"/>	>	[A,	<input type="text"/>]
<input type="text"/>	;	<input type="text"/>	>	[B,	<input type="text"/>]

5. Punkt kontrolny wyrównujący

Programista działa na strumieniu zdarzeń

```
DataStream<SensorData> inputSource = env.addSource(...);
```

gdzie klasa `SensorData` zdefiniowana jest następująco

```
public class SensorData {  
    private Integer value;  
    private String sensor;  
    ... wymagane konstruktory, getters i setters  
}
```

Zdarzenia w strumieniu mają przypisane na wejściu etykiety czasowe zdarzeń.
i na tych tylko etykietach opiera się system przetwarzania strumieni danych.

Program, który został napisany przez programistę

- odbiera dane z tematu Kafki
- filtruje dane, odrzucając błędne (nie dające się sprowadzić do obiektów `SensorData`)
- partycjonuje dane na podstawie nazwy sensora
- wylicza dla każdego sensora sumy wartości
- wysyła dane do nieistotnego w naszych rozważaniach ujścia

Szczegóły implementacyjne:

- okno *tumbling* o długości 4 (okna mają domknięte granice początkowe i otwarte granice końcowe)
- wyzwalacz natychmiastowy
- zawartość kolejnych tafli w ramach tego samego okna oparta jest na podejściu akumulacyjnym
- dopuszczalne opóźnienie wynosi 3 (watermark)

Szczegóły techniczne:

- operator źródła ma poziom zrównoleglenia 1 - oznaczmy go S
- operator filtra (osadzony zaraz po operatorze źródła, mający na celu odrzucanie danych niepoprawnych) ma poziom zrównoleglenia 1 - oznaczmy go F
- operator, który dokonuje agregacji ma poziom zrównoleglenia 2 - oznaczmy jego składowe jako RA i RB
 - do RA trafiają dane dotyczące sensora A
 - do RB trafiają dane dotyczące sensora B
- operatorem ujścia nie będziemy się zajmowali w naszych rozważaniach

Dane wejściowe mają następujący format:

```
I;etykieta_czasowa_zdarzenia> [sensor, value]
```

i są następujące:

```
I;00> [A, 08]  
I;01> [B, 05]  
I;06> [B, 03]  
I;02> [A, 08]  
I;03> [B, 03]  
I;04> [A, 09]  
I;07> [A, 04]  
I;09> [B, 04]  
I;10> [A, 04]  
I;05> [B, 05]  
>>>bariera<<<  
I;11> [A, 07]  
I;08> [B, 01]
```

Założ, że

- operator źródła generuje znaczniki watermark
- na końcu tego strumienia wysyłany jest znacznik watermark o wartości +nieskończoność
- źródłowy temat Kafki ma jedną partycję i nie było w nim wcześniej innych danych, a zatem pierwsze zdarzenie w naszych danych wejściowych miało offset 0
- system przetwarzania strumieni danych korzysta z mechanizmu punktu kontrolnego **wyrównującego**
- bariera punktu kontrolnego została wysłana do operatora źródła natychmiast po pojawieniu się zdarzenia **I;05>** [B, 05]
- w momencie pojawienia się bariery
 - w każdym buforze wejściowym i każdym buforze wyjściowym było jedno zdarzenie
 - w kanałach komunikacyjnych pomiędzy operatorami nie było żadnych zdarzeń
 - nie było żadnych zdarzeń aktualnie przetwarzanych przez operatory
- jeśli dla danego typu punktu kontrolnego bariera ma prawo wyprzedzać zdarzenia (w kanałach i buforach) to w naszym przypadku robi to natychmiast, czyli przebiega przez system szybciej zanim zajdą jakiegokolwiek zmiany w zawartościach kanałów i/lub buforów.

Odpowiedz na kilka poniższych pytań dotyczące zawartości punktu kontrolnego zapisanych przez poszczególne operatory

Elementy **operatora źródła S:**

- ostatni offset: - wpisz 0 jeśli nie jest zapamiętywany
- ostatni z wysłanych znaczników watermark: - wpisz 0 jeśli nie jest zapamiętywany
- zawartość bufora wyjściowego (OUT):

Elementy **operatora filtra F:**

- ostatni offset: - wpisz 0 jeśli nie jest zapamiętywany
- ostatni z wysłanych znaczników watermark: - wpisz 0 jeśli nie jest zapamiętywany
- zawartość bufora wejściowe (IN):
- zawartość bufora wyjściowego (OUT):

Elementy **operatorów agregacji:**

RA:

- ostatni offset: - wpisz 0 jeśli nie jest zapamiętywany
- zawartość bufora wejściowe (IN):
- liczba rekordów związanych ze stanem poszczególnych okien:
- wartość sumy w rekordzie dla najstarszego z okien: - wpisz 0 jeśli nie ma żadnych zapamiętanych stanów okien

RB:

- ostatni offset: - wpisz 0 jeśli nie jest zapamiętywany
- zawartość bufora wejściowe (IN):
- liczba rekordów związanych ze stanem poszczególnych okien:
- wartość sumy w rekordzie dla najstarszego z okien: - wpisz 0 jeśli nie ma żadnych zapamiętanych stanów okien

6. Punkt kontrolny niewyrównujący

Programista działa na strumieniu zdarzeń

```
DataStream<SensorData> inputSource = env.addSource(...);
```

gdzie klasa `SensorData` zdefiniowana jest następująco

```
public class SensorData {  
    private Integer value;  
    private String sensor;  
    ... wymagane konstruktory, gettery i settery  
}
```

Zdarzenia w strumieniu mają przypisane na wejściu etykiety czasowe zdarzeń.
i na tych tylko etykietach opiera się system przetwarzania strumieni danych.

Program, który został napisany przez programistę

- odbiera dane z tematu Kafka
- filtruje dane, odrzucając błędne (nie dające się sprowadzić do obiektów `SensorData`)
- partycjonuje dane na podstawie nazwy sensora
- wylicza dla każdego sensora sumy wartości
- wysyła dane do nieistotnego w naszych rozważaniach ujścia

Szczegóły implementacyjne:

- okno *tumbling* o długości 4 (okna mają domknięte granice początkowe i otwarte granice końcowe)
- wyzwalacz natychmiastowy
- zawartość kolejnych tafli w ramach tego samego okna oparta jest na podejściu akumulacyjnym
- dopuszczalne opóźnienie wynosi 3 (watermark)

Szczegóły techniczne:

- operator źródła ma poziom zrównoleglenia 1 - oznaczmy go S
- operator filtra (osadzony zaraz po operatorze źródła, mający na celu odrzucanie danych niepoprawnych) ma poziom zrównoleglenia 1 - oznaczmy go F
- operator, który dokonuje agregacji ma poziom zrównoleglenia 2 - oznaczmy jego składowe jako RA i RB
 - do RA trafiają dane dotyczące sensora A
 - do RB trafiają dane dotyczące sensora B
- operatorem ujścia nie będziemy się zajmowali w naszych rozważaniach

Dane wejściowe mają następujący format:

```
I;etykieta_czasowa_zdarzenia> [sensor, value]
```

i są następujące:

```
I;00> [A, 02]  
I;01> [A, 07]  
I;03> [B, 02]  
I;02> [A, 04]  
I;07> [A, 09]  
I;04> [A, 07]  
I;09> [A, 02]  
I;05> [A, 06]  
I;11> [A, 09]  
I;08> [A, 09]  
I;10> [B, 03]  
>>>bariera<<<  
I;06> [A, 06]
```


Założ, że

- operator źródła generuje znaczniki watermark
- na końcu tego strumienia wysyłany jest znacznik watermark o wartości +nieskończoność
- źródłowy temat Kafki ma jedną partycję i nie było w nim wcześniej innych danych, a zatem pierwsze zdarzenie w naszych danych wejściowych miało offset 0
- system przetwarzania strumieni danych korzysta z mechanizmu punktu kontrolnego **niewyrównującego**
- bariera punktu kontrolnego została wysłana do operatora źródła w podanym powyżej miejscu
- w momencie pojawienia się bariery
 - w każdym buforze wejściowym i każdym buforze wyjściowym było jedno zdarzenie
 - w kanałach komunikacyjnych pomiędzy operatorami nie było żadnych zdarzeń
 - nie było żadnych zdarzeń aktualnie przetwarzanych przez operatory
- jeśli dla danego typu punktu kontrolnego bariera ma prawo wyprzedzać zdarzenia (w kanałach i buforach) to w naszym przypadku robi to natychmiast, czyli przebiega przez system szybciej zanim zajdą jakiejkolwiek zmiany w zawartościach kanałów i/lub buforów.

Odpowiedz na kilka poniższych pytań dotyczące zawartości punktu kontrolnego zapisanych przez poszczególne operatory

Elementy **operatora źródła S:**

- ostatni offset: - wpisz 0 jeśli nie jest zapamiętywany
- ostatni z wysłanych znaczników watermark: - wpisz 0 jeśli nie jest zapamiętywany
- zawartość bufora wyjściowego (OUT):

Elementy **operatora filtra F:**

- ostatni offset: - wpisz 0 jeśli nie jest zapamiętywany
- ostatni z wysłanych znaczników watermark: - wpisz 0 jeśli nie jest zapamiętywany
- zawartość bufora wejściowego (IN):
- zawartość bufora wyjściowego (OUT):

Elementy **operatorów agregacji:**

RA:

- ostatni offset: - wpisz 0 jeśli nie jest zapamiętywany
- zawartość bufora wejściowego (IN):
- liczba rekordów związanych ze stanem poszczególnych okien:
- wartość sumy w rekordzie dla najstarszego z okien: - wpisz 0 jeśli nie ma żadnych zapamiętanych stanów okien

RB:

- ostatni offset: - wpisz 0 jeśli nie jest zapamiętywany
- zawartość bufora wejściowego (IN):
- liczba rekordów związanych ze stanem poszczególnych okien:
- wartość sumy w rekordzie dla najstarszego z okien: - wpisz 0 jeśli nie ma żadnych zapamiętanych stanów okien