

Przetwarzanie strumieni danych w systemach Big Data

część 3 – czas

Krzysztof Jankiewicz

Plan

- Czas – dwie (trzy) domeny
- Okna czasowe
- Obsługa zdarzeń nieuporządkowanych
- Wyzwalacze
- Utrzymywanie stanu okien
- Zawartość wyniku

Animacje i część rysunków:

“*Streaming Systems*” by Tyler Akidau, Slava Chernyak, and Reuven Lax (O’Reilly).

Copyright 2018 O’Reilly Media, Inc., 978-1-491-98387-4.

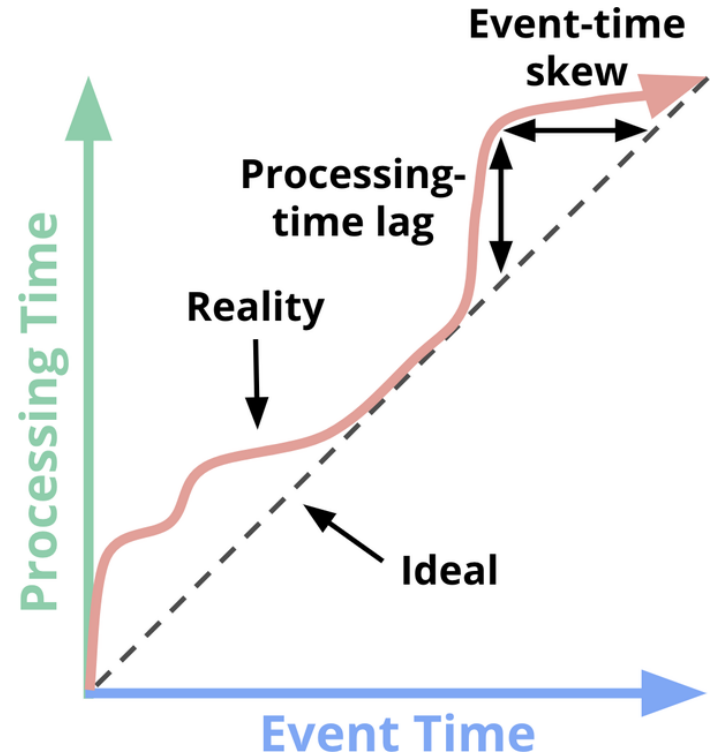
Czas – dwie (trzy) domeny

- Czas przetwarzania (*processing time*)
 - stosowany od systemów I generacji
 - nie pozwala przetwarzać danych
 - zgodnie z momentem ich rzeczywistego powstania
 - historycznych
 - nie wystarczający do wielu przypadków użycia
- Czas zdarzeń (*event time*) – rzeczywisty czas powstania zdarzenia
 - stosowany od systemów III generacji
 - pozwala na implementację architektury Kappa (dane historyczne)
 - powinien uwzględniać możliwe nieuporządkowanie danych
- Czas pozyskania (*ingestion time*) – czas pojawienia się zdarzenia w ogólnie rozumianym systemie
 - porównywalny do czasu zdarzeń (przy założeniu, że zdarzenia docierają do systemu w przybliżeniu w czasie rzeczywistym, bez opóźnień)
 - wymagania dla systemów przetwarzania analogiczne jak dla czasu zdarzeń
 - często nie jest wyodrębniany jako oddzielna domena czasu



Czas przetwarzania a czas zdarzeń

- Świat nie jest idealny – czas przetwarzania \neq czas zdarzeń
 - ograniczenia zasobów
 - oprogramowanie
 - rozproszenie przetwarzania
 - charakterystyka danych
- Opóźnienie w przetwarzaniu (*processing time lag*) – opóźnienie pomiędzy czasem wystąpienia zdarzenia a momentem jego przetwarzania
- Odchylenie czasu zdarzenia (*event-time skew*) – mówi jak daleko od ideału (w stosunku do czasu zdarzenia) jest przetwarzanie danych



Wskaż na powyższym wykresie fragmenty czerwonej linii, które odpowiadają poniższym sytuacjom:

System, którego połączenie ze światem zewnętrznym znacząco ograniczyło przepustowość

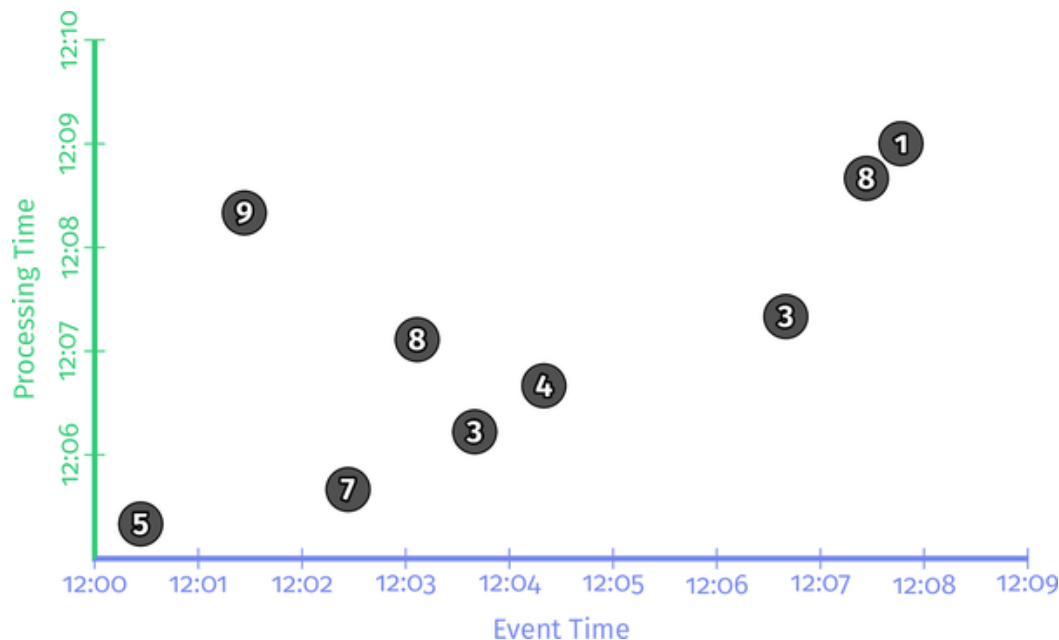
Pasażerowie samolotu, po którego wylądowaniu wyłączają tryb samolotowy w swoich urządzeniach, wysyłając i odbierając w krótkim okresie czasu wszystko, co wydarzyło się przez ostatnich kilka godzin.

Przykład strumienia danych

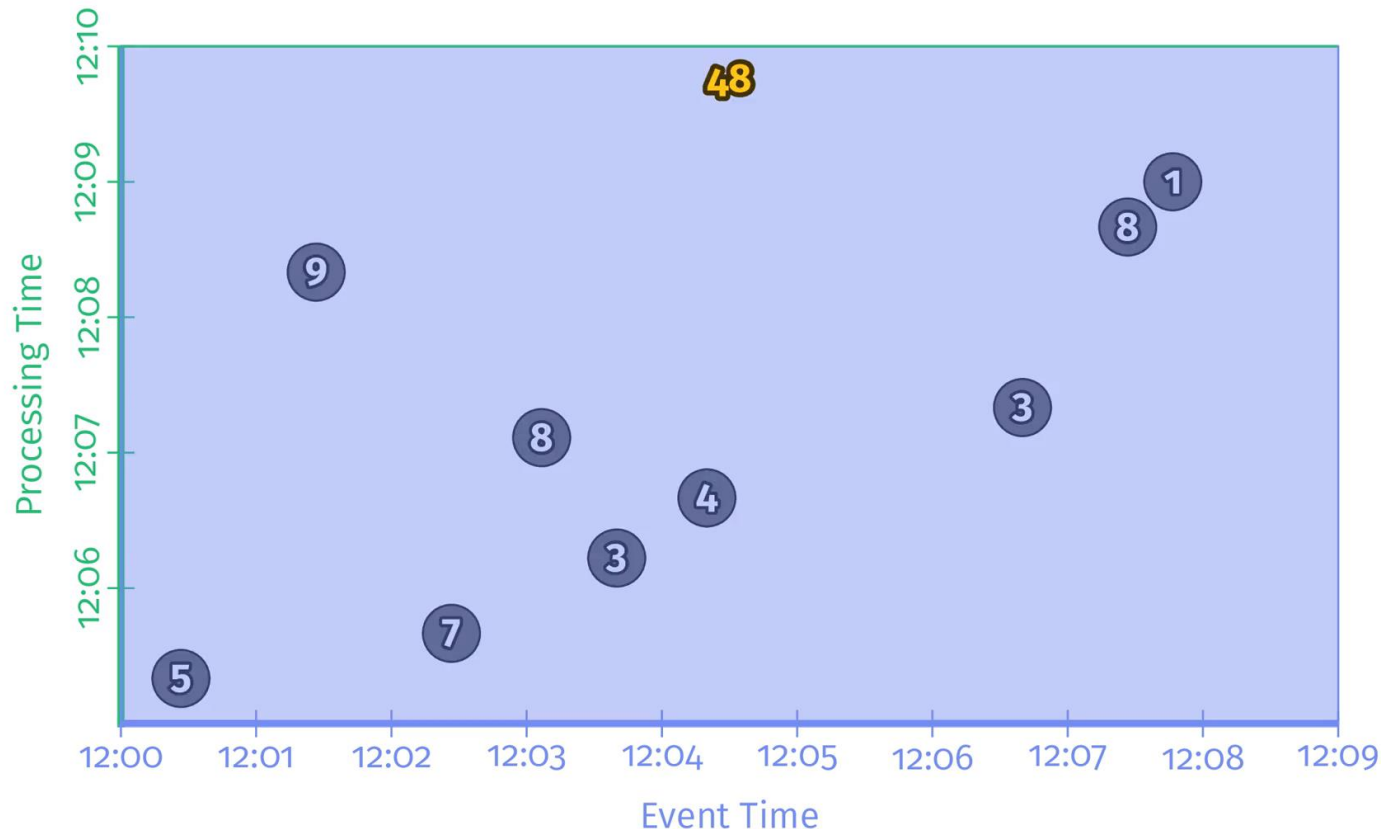
Name	Team	Score	EventTime	ProcTime
Severus Snape	Gryffindor	5	12:00:26	12:05:19
Syriusz Black	Gryffindor	7	12:02:26	12:05:39
Hermiona Granger	Gryffindor	3	12:03:39	12:06:13
Albus Dumbledore	Gryffindor	4	12:04:19	12:06:39
Severus Snape	Gryffindor	8	12:03:06	12:07:06
Rubeus Hagrid	Gryffindor	3	12:06:39	12:07:19
Harry Potter	Gryffindor	9	12:01:26	12:08:19
Minerwa McGonagall	Gryffindor	8	12:07:26	12:08:39
Rubeus Hagrid	Gryffindor	1	12:07:46	12:09:00

Będziemy sumowali
wyniki uzyskiwane
przez poszczególne
drużyny

Upraszczamy nasz
przykład zajmując się
tylko jedną drużyną
(jedną partycją
strumienia)



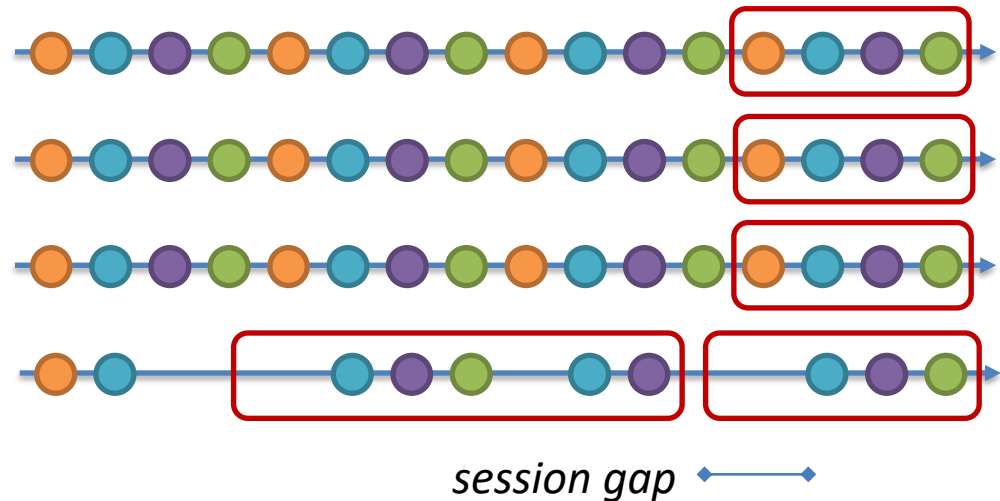
Przetwarzanie wsadowe na całości danych



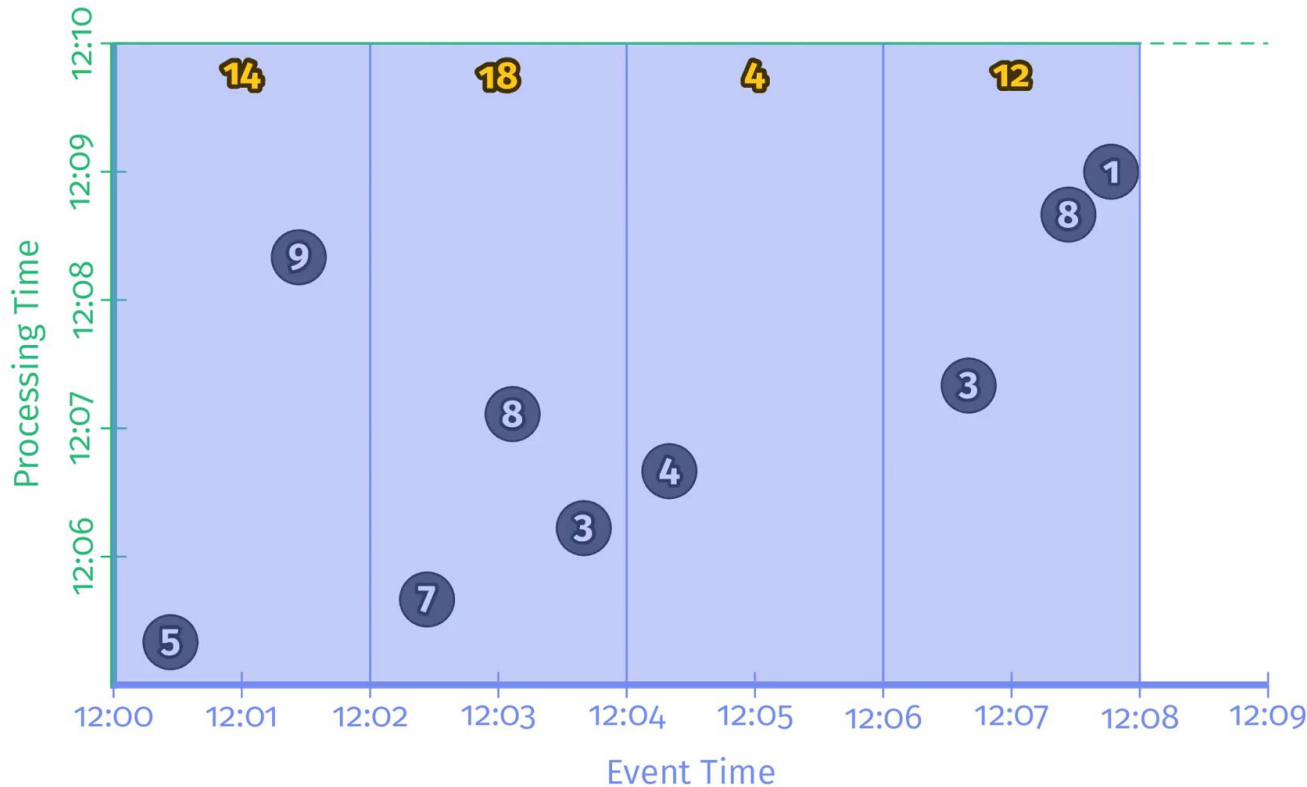
- Nieodłączny element czasu związany ze strumieniami danych powoduje, że w większości przypadków chcemy, aby otrzymywane wyniki były przypisane do określonych interwałów czasu – okien
- Okna mogą funkcjonować w różnych domenach czasu, choć głównie w domenie czasu zdarzeń

Okna

- Podział strumienia na podzbiory
- Różne typy i definicje przydziału zdarzeń do podzbiorów
- Oparte o czas (bardzo rzadko o liczbę zdarzeń)
- Typy okien
 - *Tumbling*
 - *Hopping*
 - *Sliding*
 - *Session*
 - Globalne
 - Własne
- Uproszczony podział
 - Stałe (*fixed*) – *tumbling*
 - Przesuwne (*sliding*) – *sliding, hopping*
 - Sesyjne (*sessions*) – *session*



Przetwarzanie wsadowe okien



Okna

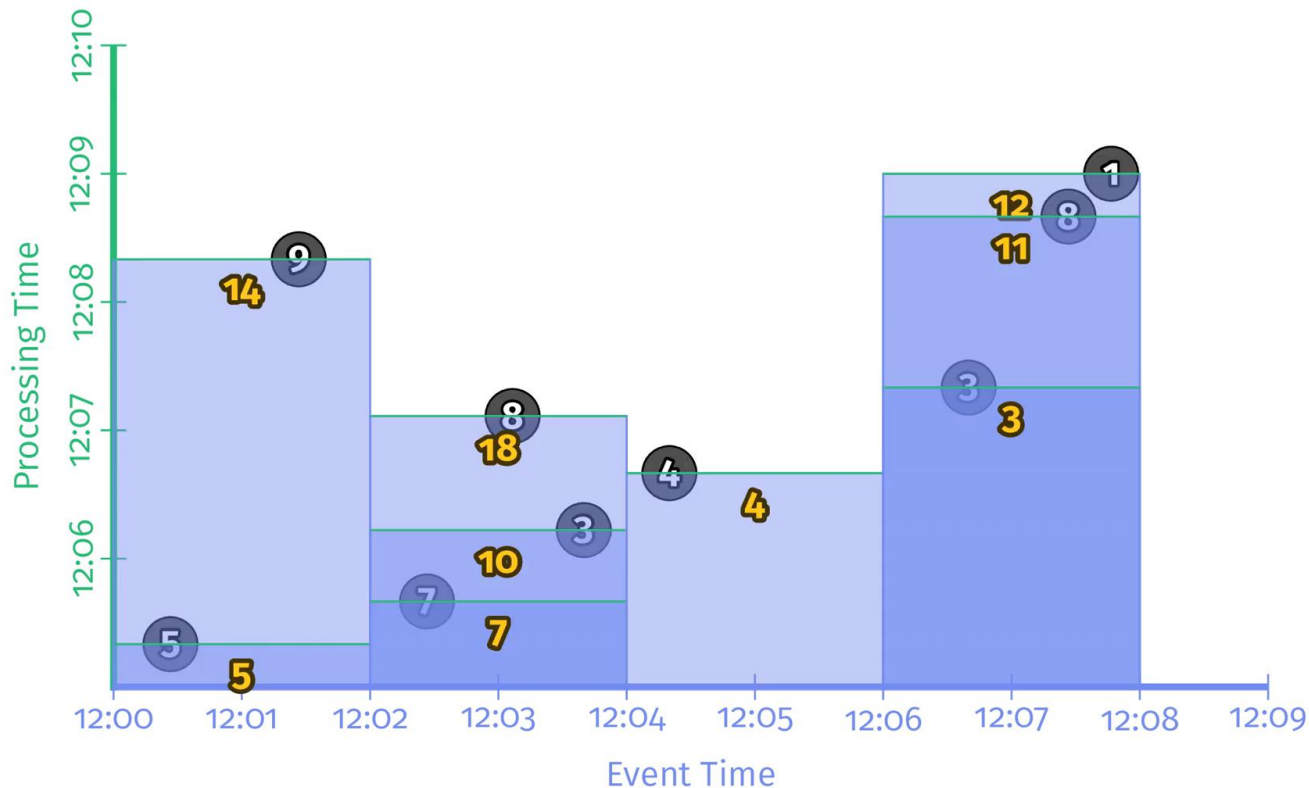
- oparte o etykiety czasowe zdarzeń
- o długości 2 minut

- W przypadku nieskończonych strumieni danych, nie jest praktycznym oczekiwanie na ich zakończenie w celu uzyskanie ostatecznego wyniku – w niektórych ;) przypadkach opóźnienie może nie być akceptowalne
- Rozwiązaniem są wyzwalacze...

Wyzwalacze

- Wyzwalacza definiują **kiedy** wynik przetwarzania okna ma być wygenerowany (ma zostać zmaterializowany)
- Każdy wynik wygenerowany dla określonego okna określany bywa taflą (*pane*)
- Wyzwalacze funkcjonują w domenie czasu przetwarzania (choć na decyzję mogą wpływać znaczniki *watermark* funkcjonujące w domenie czasu zdarzeń)
- Logika działania wyzwalaczy może być bardzo różna, jednak zazwyczaj można je zaliczyć do:
 - **wyzwalaczy powtarzalnej aktualizacji** (*repeated update triggers*) – okresowo generują tafle okna, którego zawartość ulega zmianie
 - **wyzwalaczy kompletności** (*completeness triggers*) – materializują wynik jedynie wówczas, kiedy okno można uznać za kompletne
 - będących kombinacją obu powyższych klas

Wyzwalacze powtarzalnej aktualizacji



Okna

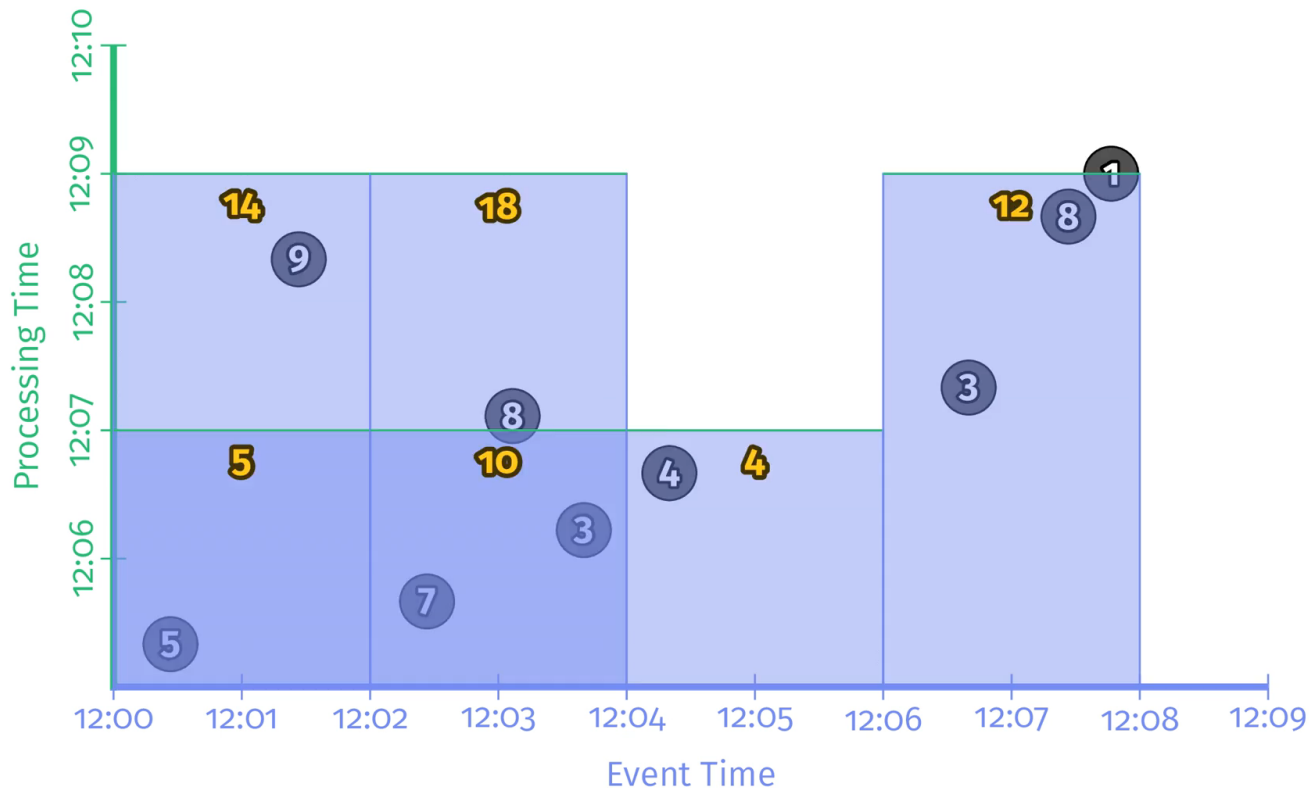
- oparte o etykiety czasowe zdarzeń
- o długości 2 minut

Wyzwalacz

- uruchamiany dla każdego zdarzenia

- Wiele taflí dla każdego okna
- Dane na wyjściu są zawsze "na bieżąco"
- Wyniki generowane są bardzo często (co w przypadku dużej ilości danych może stwarzać problemy wydajnościowe)
- Rozwiązaniem są – wyzwalacze powtarzalnej aktualizacji działające z opóźnieniem

Wyzwalacze z wyrównanym opóźnieniem



Okna

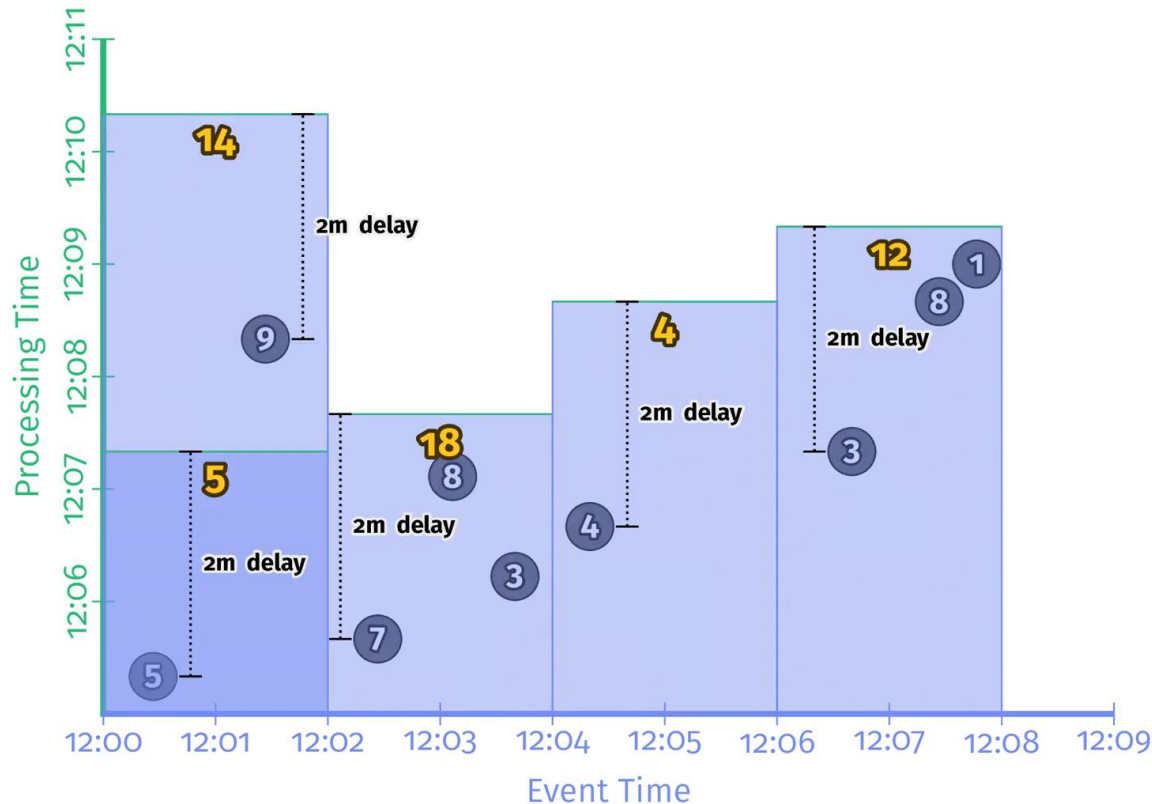
- oparte o etykiety czasowe zdarzeń
- o długości 2 minut

Wyzwalacz

- uruchamiany z wyrównanym opóźnieniem o wielkości 2 minut

- Rozwiązanie, z którym mamy do czynienia w niektórych systemach mikro-wsadowych
 - *Spark Streaming* – wielkości opóźnienia równe
 - *Spark Structured Streaming* – wielkości opóźnienia zależne od obciążenia lub równe
- Przewidywalne i regularne generowanie wyniku
- Generowanie wyników w tym samym czasie dla wszystkich okien (wszystkich partycji) prowadzi do pików obciążenia
- Rozwiązaniem są wyzwalacze z niewyrównanym opóźnieniem

Wyzwalacze z niewyrównanym opóźnieniem



Okna

- oparte o etykiety czasowe zdarzeń
- o długości 2 minut

Wyzwalacz

- uruchamiany z niewyrównanym opóźnieniem o wielkości 2 minut

- Zalety wyzwalaczy z niewyrównanym opóźnieniem dla systemów dużej skali
 - Bardziej równomierne generowanie wyników (tafli) w czasie
 - Generowanie mniejszej liczby tafli
- Korzystanie z wyzwalaczy działających z opóźnieniem skutkuje tym, że
 - dane wynikowe nie są utrzymywane "na bieżąco"
 - poprawność danych wynikowych (aktualność) nie jest w każdej chwili gwarantowana, a czas jej uzyskania nie jest do końca określony

Znaczniki *watermark*

- **Poprawność** danych wynikowych jest **gwarantowana** dla okien, które przetworzyły wszystkie swoje dane – **okien kompletnych**
- **Wiedza** dotycząca kompletności okien może być wykorzystania przez **wyzwalacze kompletności**
- Jej **źródłem** w systemach przetwarzających strumienie danych mogą być **znaczniki *watermark***
- Znaczniki *watermark* funkcjonują w domenie **czasu zdarzeń**
- Znaczniki *watermark* nigdy **nie cofają się** w czasie – ich wartości monotonicznie **rosną**
- Konceptyjnie znaczniki *watermark* można potraktować jako **funkcję**

$$f(P) \rightarrow E$$

gdzie:

P – jest czasem przetwarzania, w którym wyznaczany jest znacznik *watermark*

E – jest czasem w domenie czasu zdarzeń wskazującym, że wszystkie zdarzenia mające czas zdarzeń mniejszy niż E zostały już dostarczone

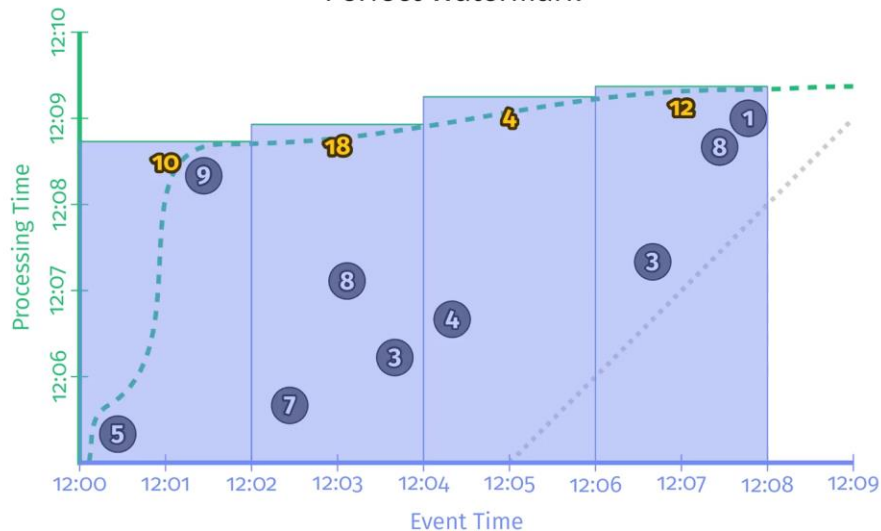
Typy znaczników *watermark*

- **Dokładne** (*perfect watermarks*) – gwarantowane
 - dostępne w sytuacji, gdy mamy dokładną (pełną) wiedzę na temat danych wejściowych
 - dane, które nie spełniają deklaracji znacznika *watermark* zdarzenia spóźnione (*late data*) – nie występują
- **Szacunkowe** (*heuristic watermarks*)
 - stosowane w znaczącej większości przypadków
 - wykorzystywane wówczas, gdy dokładna wiedza na temat danych wejściowych jest nieosiągalna
 - tworzone są na podstawie charakterystyki danych źródłowych a także wykorzystywanych do ich przetwarzania (wcześniejszych) systemów
 - nawet przy najlepszych oszacowaniach, dane spóźnione mogą pojawić się w strumieniu wejściowym

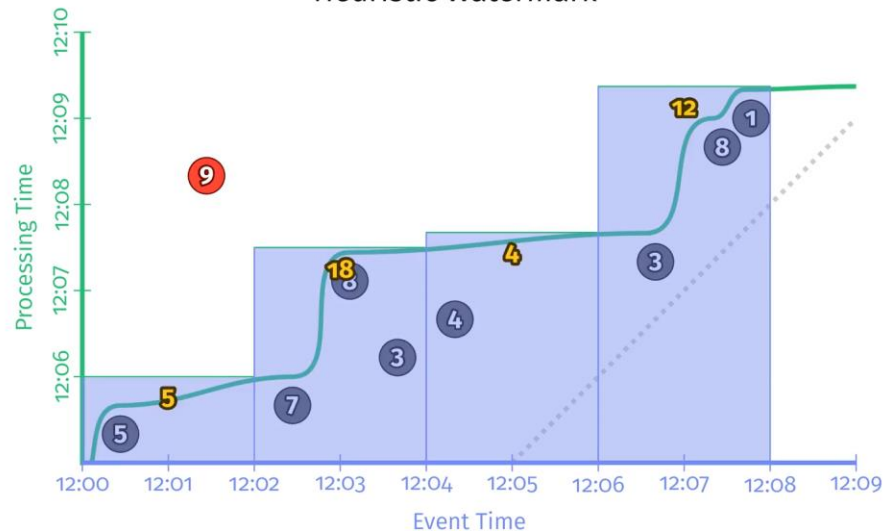
Wyzwalacze kompletności

- Mając znaczniki *watermark* możemy wykorzystać wyzwalacze, które na nich bazują.
- Wyzwalacz kompletności materializuje wynik dla okna (tworzy taflę) w momencie, gdy znacznik *watermark* przechodzi przez koniec okna

Perfect Watermark



Heuristic Watermark



Znaczniki *watermark* wykorzystywane są także przy połączeniach strumieni danych

"Dylematy" znaczników *watermark*

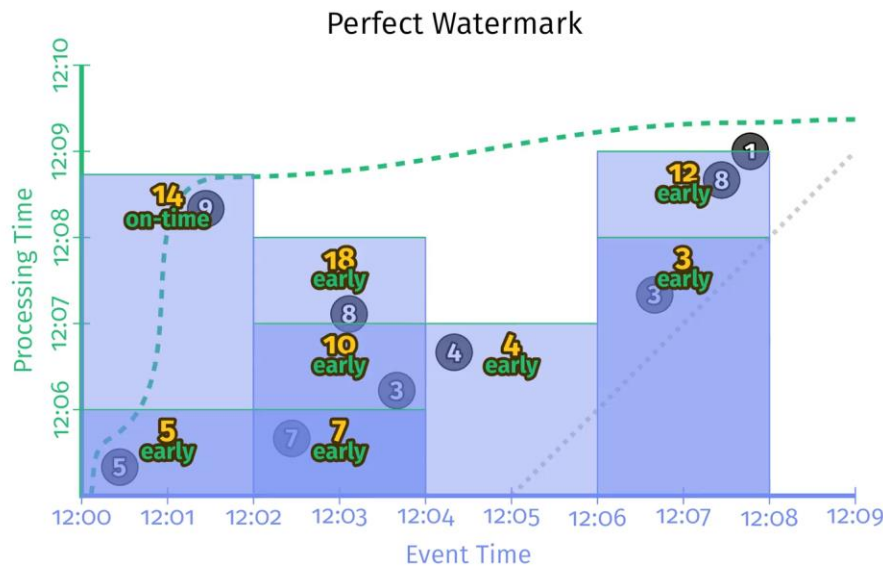
- W kontekście korzystania z wyzwalaczy kompletności znaczniki *watermark* mogą być generowane:
 - zbyt późno
 - patrz: dokładny *watermark* na naszym przykładzie – wyniki z
 - zdarzenia, które znacząco odbiegają od sytuacji idealnej (czas przetwarzania = czas zdarzeń) wstrzymują generowanie wyników dla wielu okien
 - są dobre z punktu widzenia kompletności wyników
 - bardzo negatywnie wpływają na opóźnienia
 - zbyt szybko
 - tylko w przypadku szacunkowych znaczników *watermark*
 - prowadzą do niepoprawnych rezultatów (nie obejmujących zdarzeń spóźnionych)



Jak rozwiązać ten problem?

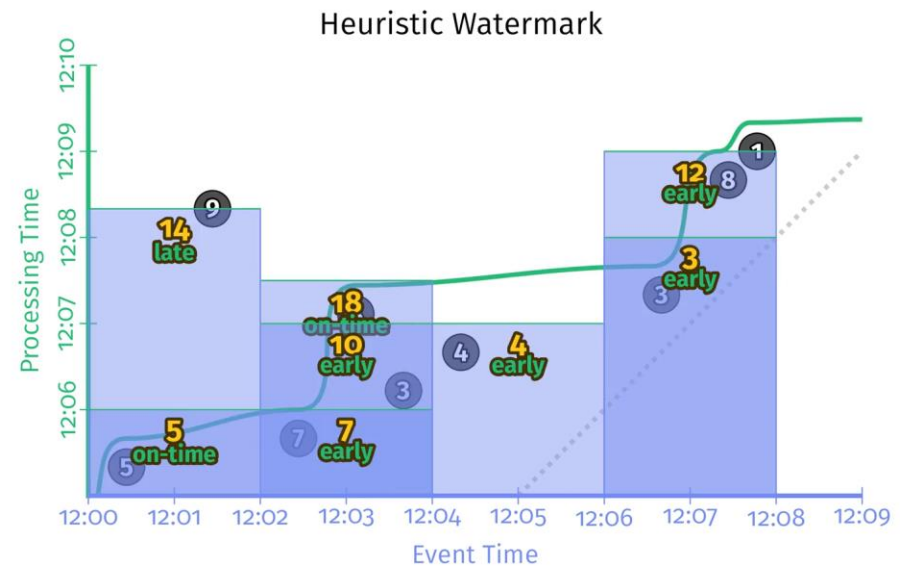
Złożone wyzwacacze



- Rozwiązaniem może być wykorzystanie złożonych typów wyzwacaczy
- Jedna z kategorii takich wyzwacaczy określana jest mianem wcześnie/na czas/późno (*Early/On-Time/Late – EOTL*). Jest ona kombinacją trzech typów wyzwacaczy
 - wcześnie – przed znacznikiem *watermark* generowane są okresowo wyniki (np. w oparciu o mechanizm wyzwacaczy z wyrównanym opóźnieniem) – zero lub więcej *wczesnych* tafli
 - na czas – znacznik *watermark* generuje kolejny wynik (w oparciu o wyzwacacz kompletności) – co najwyżej jedna tafla *na czas*
 - późno – po znaczniku *watermark* generowane są ponownie okresowo wyniki (np. wyzwacacz uruchamiany dla każdego zdarzenia) – zero lub więcej *późnych* tafli

Wyzwalacz wcześnie/na czas/późno (*EOTL*)



Perfect watermark: 
Ideal watermark: 



Heuristic watermark: 
Ideal watermark: 

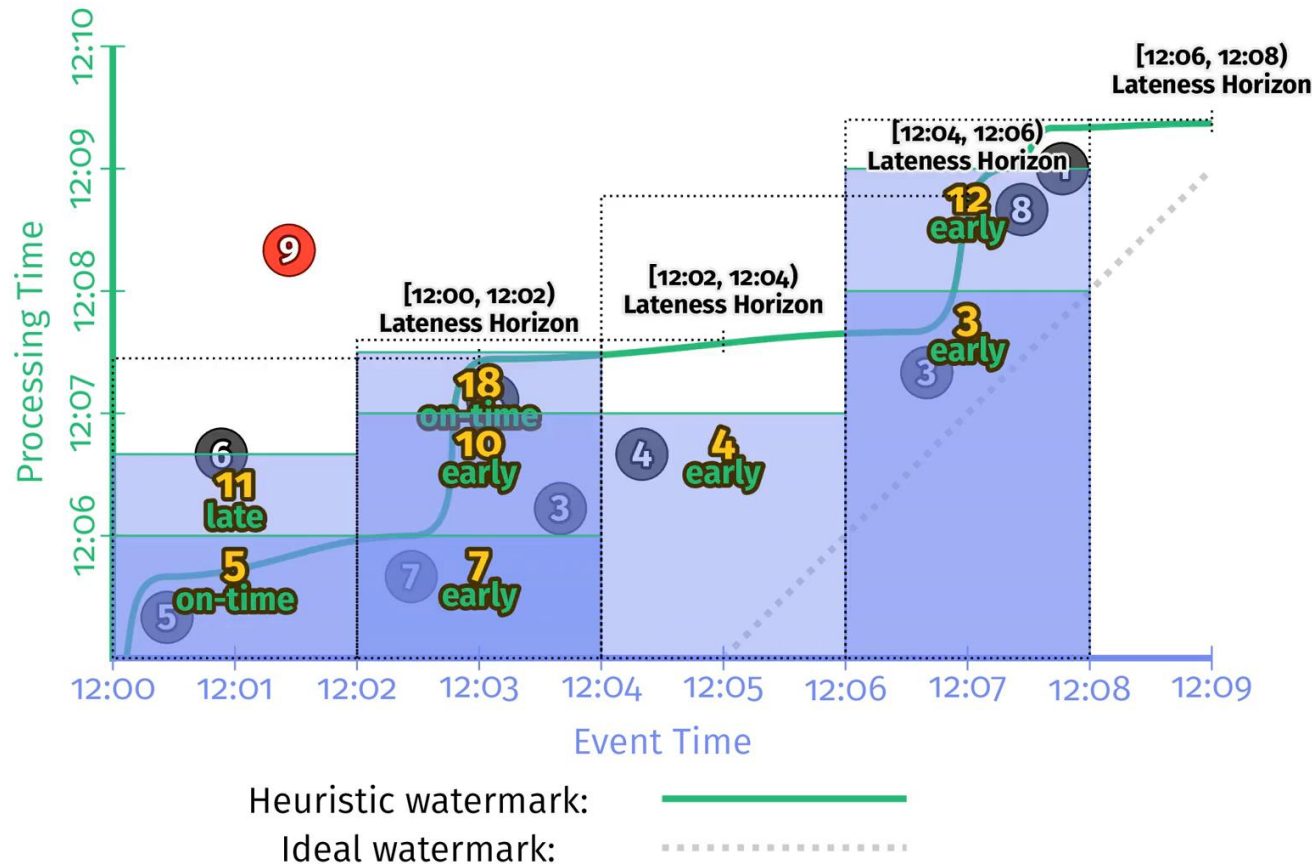
- Wyniki generowane są w sposób przybliżony dla obu typów znaczników *watermark* (wyzwalacz szacunkowy nie może być dramatycznie "zbyt szybki")
- Różnica pomiędzy typami znaczników *watermark* polega na możliwości usunięcia stanu okna
 - dokładny – dane spóźnione nigdy się nie pojawią – stan okna można usunąć
 - szacunkowy – dane spóźnione mogą się pojawić – stan okna trzeba nadal utrzymywać

Wcześnie –
wyzwalacz z
wyrównanym
opóźnieniem
co 1 minuta

Utrzymywanie stanu okien

- W praktyce nie ma możliwości przechowywania stanu okien w nieskończoność
- Konieczne jest określenie **horyzontu** dopuszczalnego **opóźnienia**, wyznaczającego dopuszczalny zakres opóźnienia danych
- Horyzont taki **pozwała na usunięcie stanu** okna
- Zdarzenia, które go **przekroczą** mogą zostać
 - **usunięte**
 - obsłużone w inny, nie oparty na stanie okna sposób (alternatywa)
- Dzięki takiemu rozwiązaniu zasoby wykorzystywane przez dane, którymi "nie jesteśmy zainteresowani" mogą zostać zwolnione

Obsługa horyzontu dopuszczalnego opóźnienia



Okna

- oparte o etykiety czasowe zdarzeń
- o długości 2 minut

Wyzwalacz

- *EOTL*

Horyzont opóźnienia

- 1 minuta opóźnienia w stosunku do znacznika *watermark*

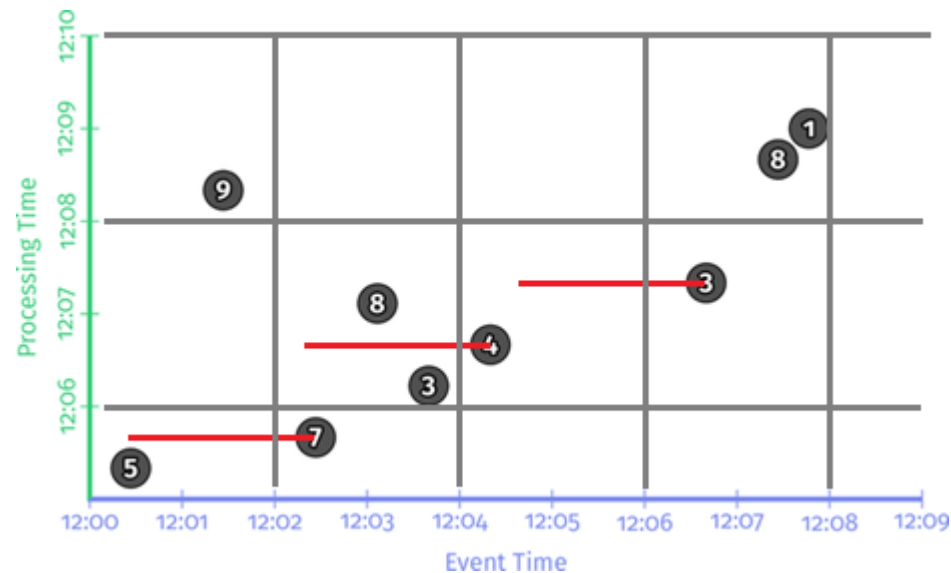
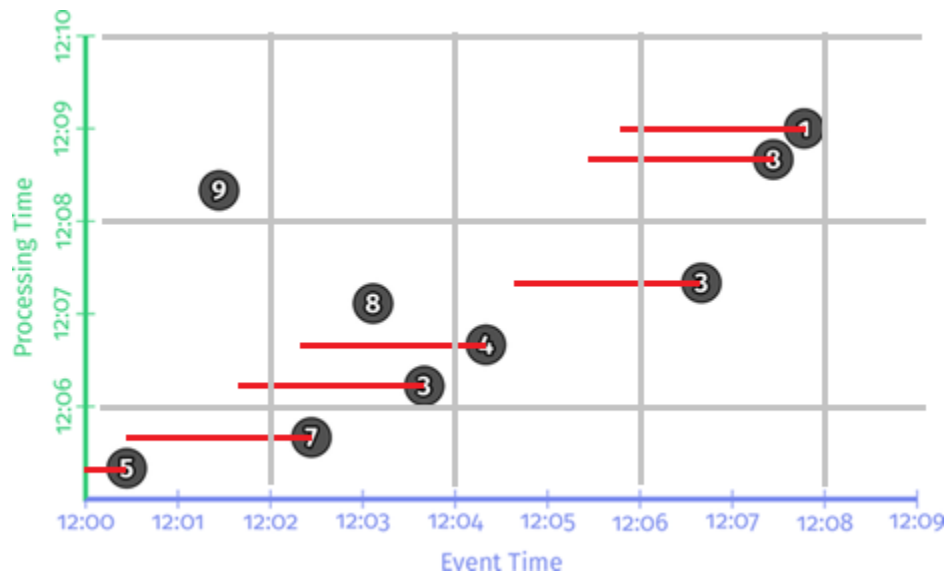
- Horyzont dopuszczalnego opóźnienia nie jest potrzebny gdy:
 - korzystamy z dokładnego znacznika *watermark*
 - dokonujemy obliczeń niezwiązanych z oknami – np. agregacji dla druzyn w całym okresie czasu – zakładamy w takim przypadku, że liczba kluczy jest ograniczona (w odróżnieniu od liczby okien)

Niskie (*low*) i wysokie (*high*) watermark

- Znaczniki *watermark* wyznaczane są w oparciu o zaobserwowane znaczniki czasu zdarzeń
 - **niskie** – na **najstarszych** znacznikach
 - **wysokie** – na **najnowszych** znacznikach

Przykład:

- szacunkowy *watermark* – minus 2 minuty od zaobserwowanego znacznika



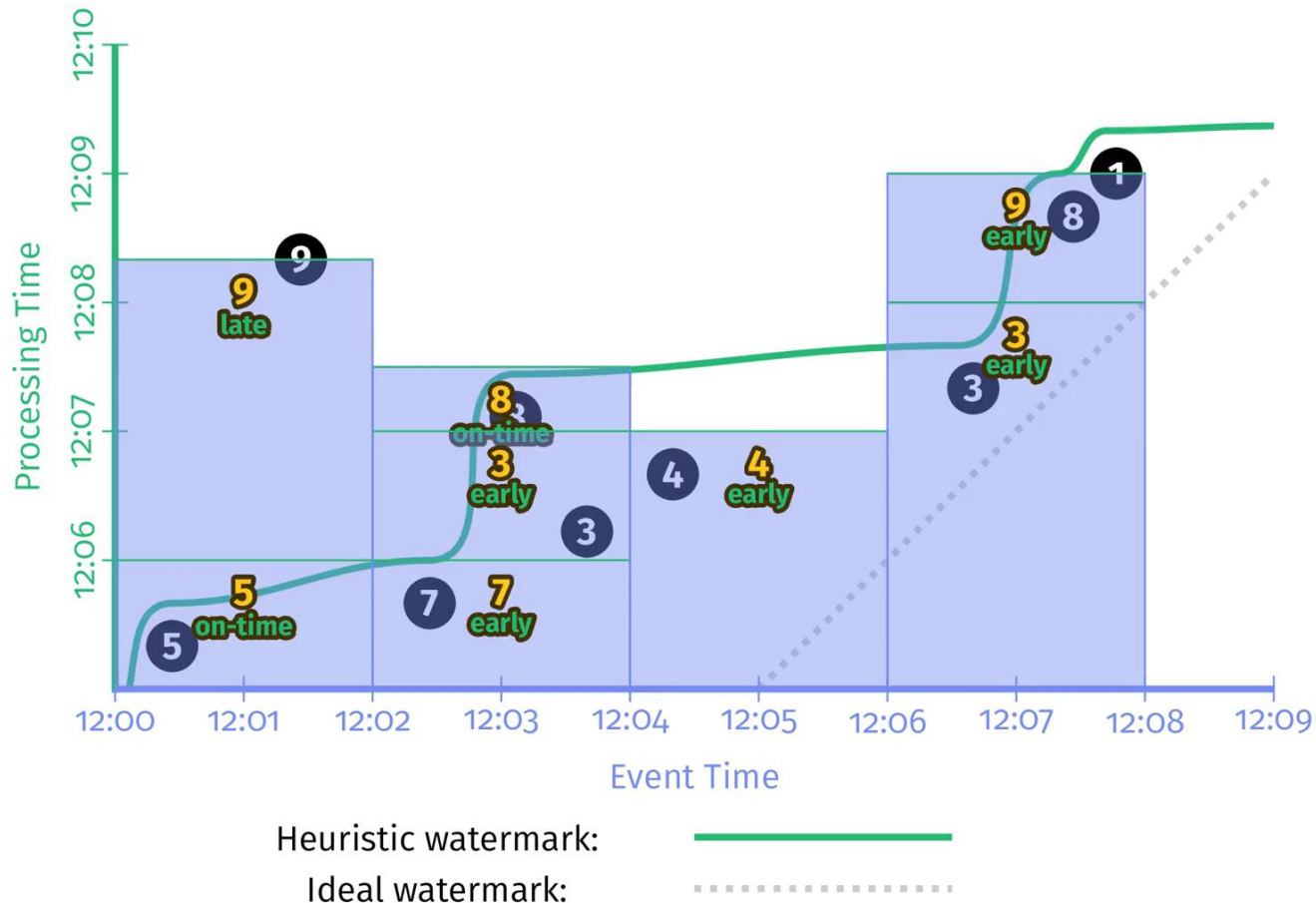
- Wysoki znacznik *watermark* stosowany jest przykładowo w *Spark Structured Streaming*
- Niskie znaczniki *watermark* stawiają na poprawność wyniku
- Wysokie znaczniki *watermark* koncentrują się na małym opóźnieniu (mając większą tendencję do występowania zdarzeń spóźnionych)

Zawartość wyniku (tafli)

- Wiemy już w jaki sposób zarządzamy
 - czasem generowania wyników (tafli)
 - czasem usuwania stanów okien
- Pozostaje pytanie: co powinno być zawarte w kolejnych wynikach dla danego okna?
- Trzy podejścia:
 - porzucanie
 - każda tafla obejmuje tylko dane, które pojawiły się po wygenerowaniu poprzedniej tafli
 - wynik każdej tafli jest porzucany – nie jest wykorzystywany do kolejnych obliczeń
 - przydatne gdy odbiorca wyników dokonuje samodzielnie ich agregacji
 - akumulacja
 - tafla obejmuje wszystkie dane jakie pojawiły się do tej pory
 - wynik każdej tafli jest wykorzystywany do kontynuowania obliczeń dla tafli następnych
 - przydatne gdy odbiorca zastępuje wartości zaobserwowane do tej pory
 - akumulacja i wycofanie
 - tafla zawiera zarówno wynik obejmujący akumulację wartości jak i wycofane wartości poprzednie

Zawartość wyniku (tafli) – porzucanie

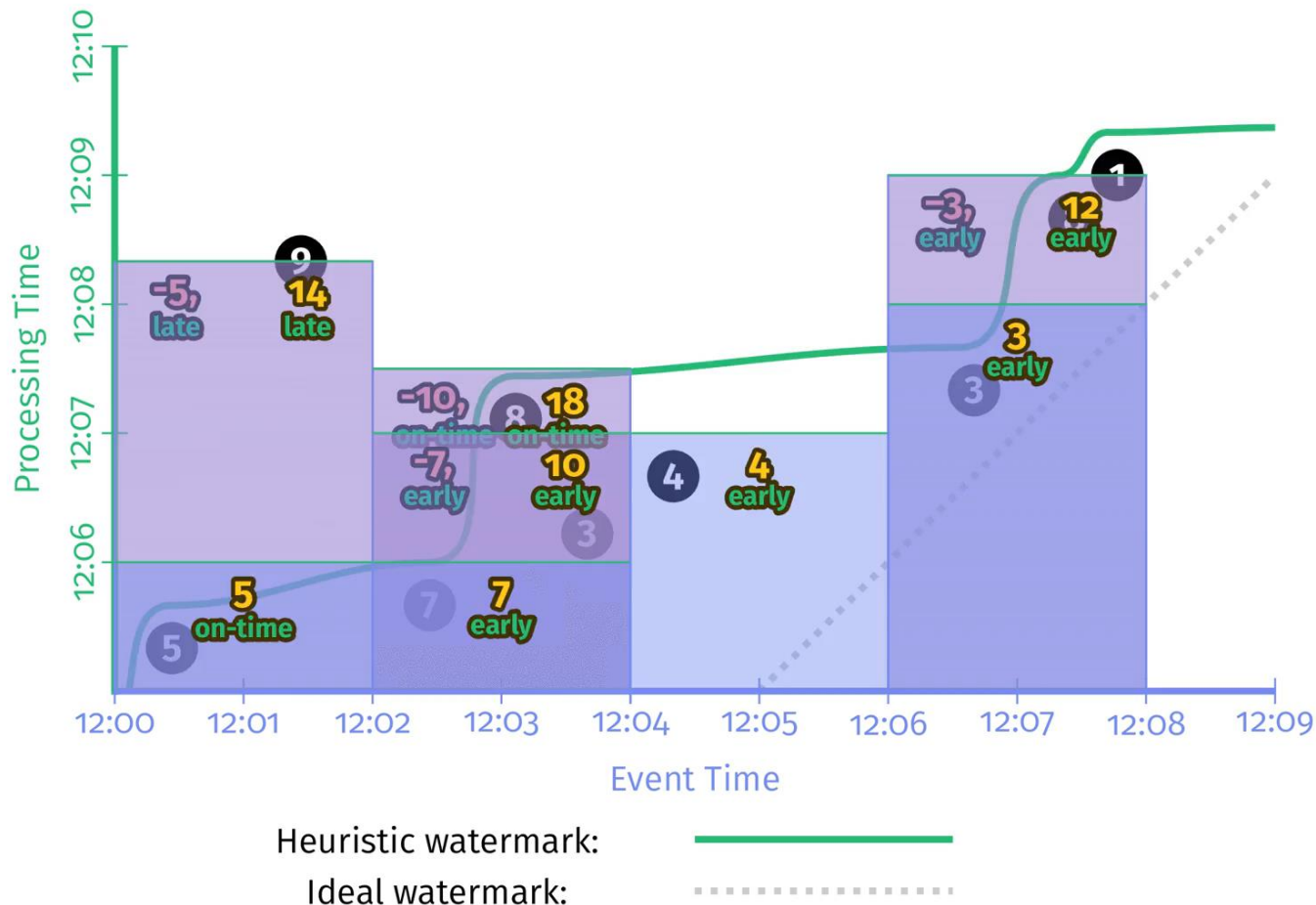
- W założeniu, korzystamy z wyzwalacza *EOTL*.



Zawartość wyniku (tafli)

akumulacja i wycofanie

- W założeniu, korzystamy z wyzwalacza *EOTL*.



Zawartość wyniku (tafli)

akumulacja i wycofanie

- Daje duże możliwości w kontekście dalszej obsługi
 - zmiana sposobu grupowania
 - obsługa dynamicznych okien (np. sesyjnych)
- Stosowane bardzo często w systemach przetwarzania strumieni danych
- W szczególności widoczne w wyższych poziomach abstrakcji (np. SQL)

Przykład

- Wynikiem przetwarzania strumieni jest to kto dla jakiej drużyny zdobył ostatnio punkty:
`select character, last(team)`
`from scores`
`group by character`
- Na podstawie wyniku odbiorca chce utrzymywać liczbę postaci grających dla każdej z drużyn
- Rozważ obsługę trzech typów wyników przetwarzania strumieni:
 - porzucanie
 - akumulacja
 - akumulacja i wycofanie

Pytania?