STUDIA INFORMATICA 2012
Volume 33 Number 2A (105)

Marcin GORAWSKI
Politechnika Śląska, Instytut Informatyki, Politechnika Wrocławska, Instytut Informatyki
Krzysztof PASTERAK
Politechnika Śląska, Instytut Informatyki

# SCHEDULERY STRUMIENIOWE W AGKPSTREAM

**Streszczenie**. Artykuł przedstawia prototypowy system strumieniowego przetwarzania danych *AGKPStream*. Główny nacisk położony jest na problem wydajnego zarządzania pracą poszczególnych komponentów przetwarzających tegoż systemu (operatorów strumieniowych). Realizowane jest to przez próbę prawidłowego doboru wybranych algorytmów szeregowania (schedulerów strumieniowych).

**Słowa kluczowe**: schedulery strumieniowe, statystyki pracy operatorów strumieniowych, strumieniowe przetwarzanie danych

# STREAM SCHEDULERS

**Summary**. The following paper introduces a prototype Data Stream Management System AGKPStream. The principal aspect of this work is to solve the problem of efficient tasking of data processing components of AGKPStream (stream operators). It is realized by experimental selection of several stream scheduling policies.

**Keywords**: stream schedulers, statistics of stream operators, stream data processing

## 1. Wstęp

Mnogość i różnorodność zastosowań w dziedzinie ogólnie pojętego przetwarzania danych doprowadziła do powstania kilku różnych paradygmatów przetwarzania danych. Najbardziej znanym przykładem jest relacyjne przetwarzanie danych, które jest wykorzystywane przede wszystkim w bazach danych ogólnego użytku. Obok innych, mniej lub bardziej podobnych sposobów postrzegania i przetwarzania danych na uwagę zasługuje model strumieniowy, którego dotyczy ten artykuł.

Znajduje on zastosowanie przede wszystkim w systemach wymagających ciągłego przetwarzania danych, które przychodzą do systemu na bieżąco i zmieniają się w sposób dynamiczny [4]. Przykładami mogą być problem monitorowania ruchu drogowego, analiza danych giełdowych lub śledzenie ruchu internetowego wraz z wykonywaniem pewnych operacji na zebranych w ten sposób danych [4]. Wszystkie te przykłady charakteryzują się ciągłym napływem danych, zmiennym tempem ich napływania oraz potrzebą generowania wyników w krótkim czasie [4].

System przetwarzania strumieniowego musi umożliwiać wykonywanie na przychodzących danych różnych operacji, zwanych dalej zapytaniami. Podobnie jak w przypadku relacyjnych baz danych zapytania te są formułowane przez użytkownika końcowego takiego systemu. W związku z tym część systemu odpowiedzialna za realizację takich zapytań musi udostępniać użytkownikowi pewien zbiór elementów funkcjonalnych, potrafiących wykonywać elementarne operacje na danych, z których to elementów użytkownik będzie w stanie zestawić gotowe zapytanie. Elementy te zwane są operatorami. Operatory połączone są ze sobą za pomocą strumieni, w których płyną dane.

Aby system przetwarzania strumieniowego mógł pracować wydajnie oraz wychodził naprzeciw zmiennym warunkom pracy, istotne jest prawidłowe zarządzanie pracą operatorów wchodzących w skład zapytania. Elementem odpowiedzialnym za to jest scheduler. Podobnie jak w przypadku systemu operacyjnego, gdzie scheduler przydziela wszystkim zarejestrowanym w nim procesom zasoby komputera, scheduler w systemie strumieniowym wywołuje każdy operator na zadany okres czasu, według określonej strategii.

Zaprojektowany i zaimplementowany autorski system przetwarzania strumieniowego *AGKPStream* został zrealizowany w Zespole Teorii Przestrzeni Danych i Algorytmów w Zakładzie Teorii Informatyki Instytutu Informatyki Wydziału AEiI Politechniki Śląskiej w ramach projektów inżynierskich [1, 2]. *AGKPStream* jest kontynuacją badań nad strumieniowym modelem przetwarzania danych z prac wcześniejszych [3, 4].

## 2. Podstawowe pojęcia

System strumieniowego przetwarzania danych składa się z pewnej liczby połączonych ze sobą operatorów, które pełnią wyspecjalizowane funkcje, razem tworząc zapytania strumieniowe. Istotnym zagadnieniem jest zbieranie podczas pracy systemu pewnych informacji dotyczących operatorów oraz strumieni, zwanych dalej statystykami. Pozwalają one na wyznaczenie w danej chwili operatora, który powinien zostać wywołany do pracy przed innymi, na przykład ze względu na dużą liczbę krotek oczekujących na przetworzenie. Jest to podstawą do wyznaczenia kolejności wywoływania operatorów i jest zasadą działania pewnej grupy schedulerów – schedulerów priorytetowych.

### 2.1. Statystyki operatorów jednowejściowych

Istnieje pięć podstawowych statystyk, które mogą być uwzględniane przy obliczaniu priorytetów [5]. Wiążą się one z czasem pracy oraz liczbą krotek przychodzących i wychodzących z punktu widzenia operatora, dla którego są one zbierane.

**Wprowadzone zostają następujące oznaczenia:**

- $n_i$ – liczba krotek wejściowych, wpływających do i-tego operatora,
- $m_i$ – liczba krotek wyjściowych, wyprodukowanych (przepuszczonych dalej) przez *i*-ty operator,
- $t_i$ – całkowity czas pracy *i*-tego operatora; czas ten zliczany jest tylko wtedy, gdy dany operator pracuje, tj. przetwarza krotkę.

**Selektywność ($s_i$)**

Jest to stosunek liczby krotek wyprodukowanych ($m_i$) do liczby krotek, które przyszły na wejście ($n_i$), i wynosi:

$$s_i = \frac{m_i}{n_i}. \tag{1}$$

**Koszt ($c_i$)**

Stanowi czas potrzebny na przetworzenie pojedynczej krotki; może być liczony jako iloraz całkowitego czasu pracy ( $t_i$ ) do liczby krotek wejściowych ( $n_i$ ) i wynosi:

$$c_i = \frac{t_i}{n_i}. \tag{2}$$

**Globalna selektywność ($S_i$)**

Selektywność grupy połączonych ze sobą operatorów, kończącej się i-tym operatorem; jest ona równa iloczynowi selektywności i-tego operatora z globalną selektywnością operatora poprzedniego. W przypadku gdy dany operator jest pierwszy w grupie, jego globalna selektywność jest równa jego selektywności. Wyraża się to wzorami:

$$\begin{aligned} S_0 &= s_0 \\ S_i &= s_i \times S_{i-1} \end{aligned} \tag{3}$$

**Globalny idealny koszt ($T_i$)**

Idealny koszt przetwarzania krotki przez wszystkie operatory połączone ze sobą w grupie kończącej się i-tym operatorem; jest on równy sumie kosztu i-tego operatora i globalnego idealnego kosztu operatora poprzedniego. W przypadku gdy dany operator jest pierwszy w grupie, jego globalny idealny koszt jest równy jego kosztowi. Wyraża się to wzorami:

$$T_0 = t_0$$
$$T_i = c_i + T_{i-1} \tag{4}$$

**Globalny średni koszt ($\overline{C}_i$)**

Średni koszt przetwarzania krotki przez wszystkie operatory połączone ze sobą w grupie kończącej się i-tym operatorem; jest on równy sumie iloczynu kosztu i-tego operatora z globalną selektywnością operatora poprzedniego i globalnym średnim kosztem operatora poprzedniego. W przypadku gdy dany operator jest pierwszy w grupie, jego globalny średni koszt jest równy jego kosztowi. Wyraża się to wzorami:

$$\overline{C}_0 = c_0$$
$$\overline{C}_i = c_i \times S_{i-1} + \overline{C}_{i-1} \tag{5}$$

Przez uwzględnienie globalnej selektywności operatora poprzedzającego miara ta jest bardziej zbliżona do rzeczywistości niż globalny idealny koszt, gdyż liczba krotek produkowanych przez każdy operator może być różna od liczby krotek przychodzących do niego [5].

### 2.2. Statystyki operatorów wielowejściowych

W przypadku operatorów wielowejściowych obliczanie statystyk staje się nieco bardziej skomplikowane [5]. Dzieje się tak w przypadku statystyk globalnych, gdyż nie można użyć bezpośrednio we wzorze np. $S_{i-1}$ ze względu na wiele operatorów poprzedzających dany operator. Rozwiązaniem może być obliczenie wyrażenia reprezentującego zbiorczą statystykę globalną wszystkich wejść, a następnie użycie jej we wspomnianych wcześniej wzorach.

**Wprowadzone zostają następujące oznaczenia:**

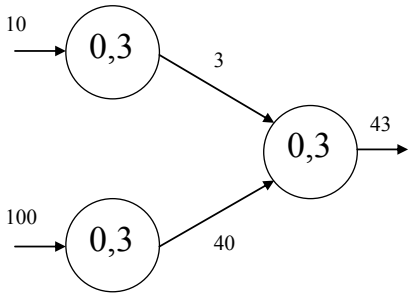
- $k$ – liczba wejść operatora,
- $n_i^j$ – liczba krotek wejściowych, wpływających na j-te wejście i-tego operatora,
- $n_i$ – całkowita liczba krotek wejściowych, wpływająca na wejście i-tego operatora, będąca sumą liczby krotek wejściowych z każdego wejścia: $n_i = \sum_{j=1}^{k} n_i^j$,
- $S_{i-1}^j$ – globalna selektywność operatora podłączonego na j-te wejście i-tego operatora,
- $T_{i-1}^j$ – globalny idealny koszt operatora podłączonego na j-te wejście i-tego operatora,
- $\overline{C}_{i-1}^j$ – globalny średni koszt operatora podłączonego na j-te wejście i-tego operatora.

Można założyć, że zbiorcza statystyka globalna jest średnią globalnych statystyk ze wszystkich wejść. Dodatkowo jest to średnia ważona, gdzie wagą jest prawdopodobieństwo

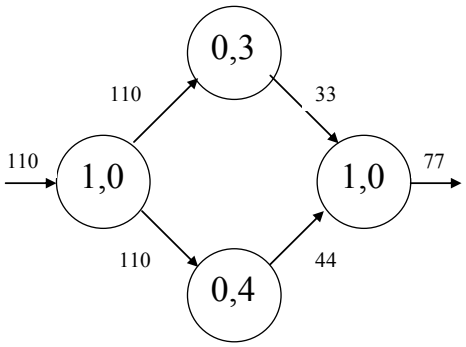
przyjścia krotki na dane wejścia, pod warunkiem że krotka wpłynęła do grupy operatorów, dla której liczona jest statystyka globalna.

**Globalna selektywność**

W przypadku obliczania globalnej selektywności dla operatorów wielowejściowych problem komplikuje się dodatkowo ze względu na fakt, iż operatory poprzedzające dany operator mogą wywodzić się z jednego wspólnego operatora bądź pochodzić z niezależnych źródeł. Problem ten ilustrują rysunek 1 (przypadek A) oraz rysunek 2 (przypadek B).




Rys. 1. Operatory poprzedzające pochodzą z niezależnych źródeł (A)
Fig. 1. Previous operators have different origin (A)




Rys. 2. Operatory poprzedzające pochodzą z tego samego źródła (B)
Fig. 2. Previous operators have the same origin (B)

Na rysunkach 1 oraz 2 przedstawione zostały dwie sytuacje, oznaczone jako A i B. W obu na wejście układów wchodzi 110 krotek. W przypadku układu A krotki wejściowe przychodzą na oba wejścia (na pierwsze wejście 10, na drugie 100), a w przypadku układu B na jedyne wejście wpływa od razu 110 krotek. Następnie krotki przechodzą przez operatory, których selektywności wynoszą odpowiednio 0,3 i 0,4. Po przetworzeniu przez te operatory liczba krotek zostaje zredukowana. Jednak w przykładzie B do operatorów o selektywności 0,3 i 0,4 jest podłączony jeden i ten sam operator, który przekazuje swoje krotki wyjściowe obu tym operatorom. Skutkiem tego łączna liczba krotek wchodzących na wspomniane operatory jest w układzie B dwukrotnie większa niż w układzie A. Na wyjściu każdego z układów pojawia się różna liczba krotek (dla układu A jest to 43, a dla B 77). Selektywność układu A (globalna selektywność ostatniego operatora) wynosi:

$$S_A = \frac{43}{10+100} \cong 0{,}39,$$

a układu B:

$$S_B = \frac{77}{110} = 0{,}7.$$

Przy formułowaniu wzorów na globalną selektywność należy rozważyć oba przypadki.

**Przypadek A**

Zachodzi wtedy, gdy operatory poprzedzające nie pochodzą od wspólnego źródła. Zbiorcza globalna selektywność operatorów wejściowych dla i-tego operatora stanowi średnią harmoniczną ważoną globalnych selektywności operatorów wejściowych:

$$S_{i-1} = \frac{n_i}{\sum_{j=1}^{k}\left(n_i^j \times \frac{1}{S_{i-1}^j}\right)} \tag{6}$$

Dla przykładu A globalna selektywność układu będzie wynosić:

$$S_A = \frac{40+3}{3\times 1/0{,}3 + 40\times 1/0{,}4} = \frac{43}{10+100} \cong 0{,}39.$$

**Przypadek B**

Zachodzi wtedy, gdy operatory poprzedzające pochodzą od wspólnego źródła. Zbiorcza globalna selektywność operatorów wejściowych dla i-tego operatora stanowić będzie sumę globalnych selektywności operatorów wejściowych:

$$S_{i-1} = \sum_{j=1}^{k} S_{i-1}^j \tag{7}$$

Dla przykładu z rysunku 2, globalna selektywność układu będzie wynosić:

$$S_B = 1{,}0 \times (0{,}3 + 0{,}4) \times 1{,}0 = 0{,}7.$$

W przypadku gdy do operatora podłączone są więcej niż 2 operatory wejściowe i część z nich zalicza się do przypadku jak w A, a część do tego drugiego, to należy najpierw obliczyć zbiorcze częściowe selektywności globalne dla operatorów wejściowych, pochodzących ze wspólnego źródła (obliczyć ich sumę), a następnie obliczone w ten sposób zbiorcze częściowe selektywności globalne uwzględnić w średniej ważonej harmonicznej.

**Globalny idealny koszt**

Zbiorczy globalny idealny koszt operatorów wejściowych dla i-tego operatora można obliczyć jako średnią arytmetyczną ważoną:

$$T_{i-1} = \frac{1}{n_i} \times \sum_{j=1}^{k} \left( n_i^j \times T_{i-1}^j \right) \tag{8}$$

Wykorzystując obliczony w ten sposób zbiorczy globalny idealny koszt, można posłużyć się wzorem opisanym dla operatorów jednowejściowych.

**Globalny średni koszt**

Zbiorczy globalny średni koszt operatorów wejściowych dla i-tego operatora obliczany jest w identyczny sposób jak globalny idealny koszt, jako średnia arytmetyczna ważona:

$$\overline{C}_{i-1} = \frac{1}{n_i} \times \sum_{j=1}^{k} \left( n_i^j \times \overline{C}_{i-1}^j \right) \tag{9}$$

Wykorzystując obliczony w ten sposób zbiorczy globalny średni koszt, można posłużyć się wzorem opisanym dla operatorów jednowejściowych, uwzględniając w nim zbiorczą globalną selektywność obliczoną w sposób pokazany w poprzedniej części.

### 2.3. Statystyki strumieni

W systemie strumieniowego przetwarzania danych strumienie należy rozumieć jako bufory pośredniczące w transporcie krotek pomiędzy operatorami. Z założenia system strumieniowy powinien przetwarzać krotki w jak najkrótszym czasie, nie powodując przy tym opóźnień. Zbyt duże nagromadzenie się krotek w strumieniu, spowodowane niewystarczającym tempem ich pobierania i przetwarzania przez element podłączony do ujścia strumienia, powoduje znaczne opóźnienie i ryzyko dewaluacji krotek. Dlatego istotne jest, by w miarę możliwości dbać o równomierny rozkład krotek we wszystkich strumieniach w systemie.

Statystyką charakterystyczną dla strumieni, jako kolekcji krotek, jest ich aktualny rozmiar, czyli liczba krotek przechowywanych w strumieniu. W dalszej części artykułu będzie ona oznaczana jako:

$\left| q_i^j \right|$ – rozmiar kolejki podłączonej na j-te wejście i-tego operatora.

## 3. Schedulery

Celem stosowania różnych typów schedulerów jest optymalizacja wybranych metryk, opisujących zachowanie się systemu strumieniowego przetwarzania danych[5, 2]:

- *czas odpowiedzi* – obliczany dla każdej krotki; jest to czas pobytu krotki w systemie, obliczany jako różnica pomiędzy czasem opuszczenia krotki przez system a czasem jej zarejestrowania w systemie;

- *spowolnienie* – jest miarą określającą jak bardzo dana krotka została obarczona opóźnieniem niezwiązanym z wymaganym kosztem jej przetworzenia, innymi słowy jest to iloraz czasu odpowiedzi systemu dla danej krotki i jej idealnego czasu przetwarzania;
- *ilość zajętej pamięci* – jest wyznaczana przez całkowitą liczbę krotek znajdujących się w systemie, to jest w strumieniach pomiędzy operatorami i w wewnętrznych strukturach operatorów, jeśli takowe istnieją.

Z reguły nie jest możliwa optymalizacja wszystkich metryk jednocześnie, gdyż polepszenie się jednej najczęściej skutkuje pogorszeniem się innej. Przykładowo jest to wynikiem faktu, że im wydajniejszy algorytm (pod względem czasowym) został zastosowany, tym bardziej skomplikowana struktura danych musi zostać użyta. W przeciwnym przypadku, gdy stosuje się prostą strukturę danych, zazwyczaj algorytm na niej operujący jest dość kosztowny czasowo.

W *AGKPStream* zaimplementowane schedulery można podzielić na 3 grupy:

- *scheduler FIFO* [6, 7] – scheduler ten wywołuje po kolei każdy z operatorów, znajdujących się na drodze krotki, która właśnie przyszła do systemu; jeżeli jest więcej krotek oczekujących na wykonanie, to scheduler wybiera tę krotkę, która czeka najdłużej, i wywołuje dla niej operatory aż do jej całkowitego przetworzenia;
- *scheduler Round Robin* [7] – scheduler ten wywołuje po kolei operatory na zadany, stały kwant czasu;
- *schedulery priorytetowe* [5, 6, 7] – grupa schedulerów, które – podobnie jak scheduler Round Robin – wywołują operatory na zadany kwant czasu, ale kolejność, w jakiej wybierane są operatory, jest zależna od priorytetu operatora; obliczanie priorytetów dokonuje się na podstawie statystyk zbieranych przez operatorów podczas pracy.

Dwa pierwsze schedulery należą do schedulerów prostych. Oznacza to, że przy wybieraniu operatorów kierują się stałą strategią, niezależną od aktualnych parametrów pracy. Z kolei schedulery priorytetowe (w systemie *AGKPStream* zaimplementowane zostały 4 rodzaje takich schedulerów) starają się dopasować do zmieniających się warunków panujących w systemie przez ustalanie priorytetów operatorów, zależnych od wspomnianych wcześniej warunków.

### 3.1. Scheduler FIFO

Scheduler FIFO wywołuje kolejne operatory na ścieżce nowo przybyłej do systemu krotki [6, 7]. Zasada jego działania opiera się na połączeniach bezpośrednich pomiędzy operatorami. Oznacza to, że wywołanie dowolnego z operatorów podłączonych bezpośrednio na wejścia systemu spowoduje przetworzenie krotki z wejścia przez wszystkie operatory znajdujące się na ścieżce krotki aż do dotarcia tej krotki na wyjście systemu. Każdy operator po przetworzeniu

krotki nie wysyła jej do bufora (tak jak to się dzieje zazwyczaj), ale wywołuje podłączony na swoje wyjście następny operator (operatory) w celu dalszego przetwarzania krotki. Za każdym razem zadaniem schedulera FIFO jest wybranie tego operatora początkowego, dla którego krotka oczekująca na przetworzenie czekała najdłużej (ma najmniejszy znacznik czasowy).

Algorytm pracy schedulera FIFO przedstawia następujący pseudokod:

```
// przy inicjalizacji, tylko raz
for (strumień : wszystkie_strumienie_w_systemie) {
   strumień.zmień_na_połączenie_bezpośrednie
}

// cyklicznie, przy każdym wywołaniu
wybrane_wejście = minimum_względem_znacznika_czasu(wszystkie_wejścia)
wybrany_operator = wybrane_wejście.podłączony_operator
wybrany_operator.pracuj()
```

Wywołanie wybranego operatora do pracy pociągnie za sobą kolejne wywołania wszystkich operatorów, znajdujących się na ścieżce przetwarzania, dla najstarszej z oczekujących krotek. Scheduler ten wywołuje każdy z operatorów tylko dla jednej krotki. Jego zaletą jest szybki czas reakcji [2], do wad z kolei należy problem powstały przy obliczaniu statystyk operatorów. Problem ów wiąże się z faktem, że przy wzajemnym wywoływaniu się operatorów zaburzony zostaje pomiar czasu pracy tych operatorów (ponieważ uwzględnione są wszystkie czasy pracy operatorów wywołanych przez pierwszy operator). Jednakże scheduler ten nie został zaprojektowany do pracy na podstawie zliczania statystyk, tak więc z punktu widzenia poprawności przetwarzania danych działa on prawidłowo.

### 3.2. Scheduler Round Robin

Scheduler Round Robin, zwany też schedulerem „krążący żeton”, wywołuje po kolei każdy z operatorów [7]. Kolejność, w jakiej wywoływane są operatory, jest stała i ustalana na początku. Wywołanie operatora może być jednokrotne lub też wielokrotne, przez stały dla każdego operatora okres. Scheduler ten nie uwzględnia żadnych warunków zewnętrznych, takich jak różne wymagania różnych operatorów. Istnieje kilka podejść związanych z obsługą operatorów, które w danym czasie nie mają jeszcze (albo już) krotek do przetworzenia. Można wyróżnić dwie takie sytuacje:

- przy przejściu do nowego operatora nie ma on oczekujących krotek,
- operator przetworzył wszystkie krotki przed upływem czasu przeznaczonego na pracę.

W pierwszym przypadku istnieją dwie różne drogi działania. Z jednej strony można po kolei szukać operatora, który ma jakiekolwiek krotki do przetworzenia, i po znalezieniu wywoływać go do pracy w normalny sposób. Z drugiej jednak strony można próbować wywoływać operator, korzystając z założenia, że krotki napłyną w najbliższym czasie, nieprzekraczającym interwału przeznaczonego na jeden operator.

W drugim przypadku jest podobnie. Gdy operator przetworzy wszystkie krotki ze strumienia podłączonego na jego wejście, można przejść do następnego operatora, skracając wspomniany interwał czasu, albo wywoływać operator w oczekiwaniu na krotki, które mogą się jeszcze pojawić.

W obu wspomnianych przypadkach wywołanie operatora, który nie ma krotek do przetworzenia, jest równoznaczne z wykonaniem niepotrzebnej, pustej instrukcji. System *AGKP-Stream* jest systemem wielowątkowym, co oznacza, że moduły wejściowe pracują w odseparowaniu od operatorów. Może więc dochodzić do sytuacji, w której operatory po przetworzeniu wszystkich krotek mogą się spodziewać, że w najbliższej przyszłości pojawi się nowa krotka. W związku z tym zdecydowano się na rozwiązanie, w którym scheduler Round Robin wywołuje każdy operator przez cały przeznaczony na to czas, bez względu na chwilową nieobecność krotek w systemie.

Algorytm pracy schedulera Round Robin przedstawia następujący pseudokod:

```
wybrany_operator = kolekcja_operatorów.get(aktualny_indeks)

aktualny_indeks = aktualny_indeks + 1
if (aktualny_indeks >= kolekcja_operatorów.rozmiar) {
   aktualny_indeks = 0
}

// jeśli pojedyncze wywoływanie operatorów
if (tryb_prosty) {
   wybrany_operator.pracuj()
}

else for (interwał_czasu) {
   wybrany_operator.pracuj()
}
```

Scheduler Round Robin przez wywoływanie każdego operatora w ten sam sposób nie uwzględnia zmiennych warunków pracy w systemie. Niektóre operatory wymagają częstszego wywoływania niż inne, ponieważ częściej dostają krotki do przetworzenia. Jednak, algorytm tego schedulera jest stosunkowo niezłożony, co także może mieć znaczenie w niektórych przypadkach.

### 3.3. Schedulery priorytetowe

Schedulery priorytetowe to grupa schedulerów, które przydzielają zarządzanym przez siebie operatorom priorytety, a następnie wywołują operatory w kolejności ustalonej dzięki tymże priorytetom. Ogólna zasada działania polega na wyborze operatora o najwyższym priorytecie spośród tych, które mają jakieś krotki do przetworzenia. Podobnie jak w przypadku schedulera Round Robin, po dokonaniu wyboru operatora jest on wywoływany jednokrotnie lub przez zadany czas. Przy obliczaniu priorytetów brane są pod uwagę statystyki operatorów lub strumieni. Zazwyczaj liczba krotek, które przetwarza operator, oraz czas jego pracy wpływają na wartość obliczonego priorytetu. Im bardziej dany operator „potrzebuje", by zo-

stać obsłużonym (na przykład z powodu dużej liczby oczekujących na przetworzenie przez niego krotek), tym wyższy dostaje priorytet.

Jako że warunki panujące w systemie ulegają zmianie podczas jego pracy, priorytety operatorów powinny być co jakiś czas uaktualniane. Jest to konieczne, gdyż mogłoby dojść do sytuacji, w której scheduler wybierałby najmniej „potrzebujący" operator, podczas gdy te, które wymagają natychmiastowej obsługi, czekałyby, powodując opóźnienia i zastoje w strumieniach. Jednak obliczanie priorytetów dla wszystkich operatorów jest zazwyczaj operacją dosyć kosztowną, więc należy znaleźć kompromis pomiędzy tymi dwoma czynnikami.

Algorytm pracy schedulerów priorytetowych przedstawia następujący pseudokod:

```
// początkowy okres, scheduler pracuje jak Round Robin, gdyż statystyki nie są
// jeszcze wiarygodne
if (początkowy_okres) {
   Round_Robin.pracuj()
}

// już po początkowym okresie pracy
else {

   // co jakiś czas obliczanie na nowo priorytetów
   // (w zależności od typu schedulera, odbywa się to w różny sposób)
   if (trzeba_obliczyć_priorytety) {
      oblicz_priorytety()
      sortuj_według_priorytetów(kolekcja_operatorów)
   }

   // wybór pierwszego operatora, który ma krotki do przetworzenia
   // (będzie miał najwyższy priorytet, bo kolekcja jest posortowana)
   wybrany_operator = wybierz_pierwszy_mający_krotki(kolekcja_operatorów)

   // dalej tak samo jak w przypadku schedulera Round Robin

   // jeśli pojedyncze wywoływanie operatorów
   if (tryb_prosty) {
      wybrany_operator.pracuj()
   }

   else for (interwał_czasu) {
      wybrany_operator.pracuj()
   }
}
```

W systemie *AGKPStream* zostały zaimplementowane 4 typy schedulerów priorytetowych. Każdy z nich oblicza priorytet w inny sposób.

**Scheduler Zachłanny [6]**

Scheduler Zachłanny, inaczej zwany też schedulerem Greedy, przy wyliczaniu priorytetu dla każdego operatora stosuje strategię zachłanną. Oznacza to, że uwzględnia on tylko lokalne statystyki wybranego operatora, takie jak jego selektywność oraz koszt. Jest to wyrażone przez następujący wzór:

$$P_i = \frac{1 - s_i}{c_i} \tag{10}$$

**Scheduler HR [5]**

Scheduler ten dla każdego operatora wyznacza jego współczynnik wyjściowy [5]. W przeciwieństwie do poprzedniego, uwzględnia on wpływ sąsiadujących operatorów na priorytet wybranego operatora. Został on zaprojektowany, by zmniejszać czas odpowiedzi krotek [5]. Wzór (11) przedstawia sposób obliczania priorytetu przez scheduler HR:

$$P_i = \frac{S_i}{\overline{C}_i} \tag{11}$$

**Scheduler HNR [5]**

Scheduler ten dla każdego operatora wyznacza jego znormalizowany współczynnik wyjściowy [5]. Normalizacja ta polega na uwzględnianiu również idealnego czasu przetwarzania krotki przez blok operatorów. Dzięki temu priorytety są przydzielane w sposób bardziej sprawiedliwy, gdyż elementy o niskim koszcie i selektywności dostają wyższy priorytet niż w przypadku stosowania schedulera HR [5]. Scheduler HNR został zaprojektowany w celu minimalizacji spowolnienia [5]. Wzór (12) przedstawia sposób obliczania priorytetu:

$$P_i = \frac{S_i}{\overline{C}_i \times T_i} \tag{12}$$

**Scheduler MTIQ [7]**

Jego działanie polega na faworyzowaniu operatorów, które mają najwięcej nieprzetworzonych krotek w strumieniach. Został on zaprojektowany w celu minimalizacji tej wielkości [7]. Priorytetem operatora jest zatem łączna liczba krotek znajdujących się we wszystkich strumieniach podłączonych na jego wejście i wyraża się wzorem:

$$P_i = \sum_{j=1}^{k} \left| q_i^j \right| \tag{13}$$

## 4. Podsumowanie

Zaprezentowane w artykule schedulery strumieniowe są podstawą działania systemu *AGKPStream*. Umożliwia on wykonywanie elementarnych operacji strumieniowych na danych. Szczegóły dotyczące operatorów strumieniowych wchodzących w skład omawianego systemu znajdują się w [1].

System może w przyszłości posłużyć jako podstawa do dalszych badań związanych z paradygmatem strumieniowego przetwarzania danych. Przez swoją modułowość umożliwia on łatwe dodawanie kolejnych implementacji zarówno schedulerów, jak i operatorów.

Najbliższym celem jest poprawa wydajności przez dopracowanie poszczególnych modułów na podstawie wniosków z przeprowadzonych eksperymentów na prototypowym systemie *AGKPStream*. Kluczowym zagadnieniem jest ograniczenie zbędnych obliczeń wykonywanych podczas pracy, a mających negatywny wpływ na osiągi systemu, i skłonienie się w stronę prostych, ale wydajnych rozwiązań.

W dalszej perspektywie leży ułatwienie formułowania zapytań oraz ogólnie rozumianej obsługi systemu z punktu widzenia użytkownika, a także lepsze wsparcie systemu dla problemu równoległości. Wspomniane zagadnienia stanowią przyczynek do dalszych badań w tej dziedzinie.

## BIBLIOGRAFIA

1. Gorawska A.: Analiza i implementacja wybranych operatorów strumieniowych. Rozprawa inżynierska, Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, Gliwice 2012.
2. Herud T., Sulski M.: Analiza implementacyjna schedulerów i operatorów dla nowego modelu przetwarzania strumieniowego. Rozprawa inżynierska, Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, Gliwice 2011.
3. Chrószcz A.: Model przetwarzania strumieniowego uwzględniający zarówno wpływ synchronizacji jak i język zapytań łączący paradygmaty języka obiektowego i deklaratywnego. Rozprawa doktorska, Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, Gliwice 2012.
4. Stochmiałek M.: Strumieniowe bazy danych STREAM: The Stanford Data Stream Management System. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2005.
5. Sharaf M. A., Chrysanthis P. K., Labrinidis A., Pruhs K.: Efficient Scheduling of Heterogeneous Continuous Queries. VLDB Endowment, University of Pittsburgh, 2006.
6. Babcock B., Babu S., Datar M., Motwani R., Thomas D.: Operator scheduling in data stream systems. The VLDB Journal, Stanford University, 2004.
7. Sutherland T. M., Pielech B., Zhu Y., Ding L. Rundensteiner E. A.: An Adaptive Multi-Objective Scheduling Selection Framework for Continuous Query Processing. IEEE Computer Society, Worcester Polytechnic Institute, 2005.

Wypłynęło do Redakcji 31 stycznia 2012 r.

**Abstract**

Today there are many different data processing paradigms. This paper focus on stream data processing, which can be applied to many applications requiring short response time of generated output data, when the nature of input data is continuous and heterogeneous.

It is important, to properly schedule all of the internal components, such as stream operators. There are various scheduling policies, each of them has a slightly different objective. Some are suitable for response time or memory usage optimizing, the other – for limitation of complexity, through simplification of algorithms.

The presented Data Stream Management System AGKPStream is a prototype system for testing purposes. In this work, the main point is to introduce basic terms connected with scheduling in stream processing system, and also to show principles of operation of all implemented stream schedulers.

**Adresy**

Marcin GORAWSKI: Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, Marcin.Gorawski@polsl.pl; Politechnika Wrocławska, Wydział Informatyki i Zarzadzania, Instytut Informatyki, ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Polska, Marcin.Gorawski@pwr.wroc.pl.

Krzysztof PASTERAK: Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, krzypas641@student.polsl.pl.