STUDIA INFORMATICA 2003
Volume 24 Number 2B (54)

Grzegorz B. PROKOPSKI
Politechnika Wrocławska, Instytut Sterowania i Techniki Systemów

# WYBRANE METODY I ALGORYTMY SZEREGOWANIA ZADAŃ W SERWERZE WWW

**Streszczenie.** Przedstawiono wymagania stawiane nowoczesnym serwerom WWW ze względu na zastosowania biznesowe. Na podstawie literatury określono specyfikę ich obciążenia związaną ze zjawiskiem długiego ogona. Stworzono listę głównych cech rozwiązań szeregowania, m.in. ze względu na rodzaj kryterium optymalizacji. Dla dwóch przykładowych rozwiązań wskazano cechy ze stworzonej listy. Zaproponowano dalsze kierunki rozwoju.

**Słowa kluczowe:** sieci komputerowe, serwery WWW, sterowanie, szeregowanie zadań.

# SELECTED METHODS AND ALGORITHMS OF TASK SCHEDULING IN A WEB SERVER

**Summary.** Requirements for modern web servers in business applications is introduced. Based on a literature the specific of their heavy tailed burden is defined. A list of main features of scheduling solutions, among others – because of the optimalisation criteria, is created. For two example solutions their features from the list are pointed out. Future research directions are proposed.

**Keywords:** computer networks, web servers, steering, tasks scheduling.

## 1. Wprowadzenie – charakterystyka obciążenia serwerów WWW

Internet stał się istotnym medium dla biznesu, sprzedaży i zakupu towarów i usług. Aplikacje e-commerce stawiają przed obsługującą je technologią serwerów WWW pewne warunki. Podstawowym i najogólniej sformułowanym wymaganiem jest zapewnienie tak szybkiej obsługi użytkowników, aby nie rezygnowali oni z zakupów czy innego użycia witryny,

a podmiot mający korzyści z działalności w Internecie – nie ponosił strat. W [5] pojęcie jakości serwisów WWW - QoWS zostało zdefiniowane jako wymaganie zapewnienia określonej części klientów jakości usługi na minimalnym przyjętym poziomie, np. czasu odpowiedzi lub przepustowości. Na postrzegany przez użytkownika czas odpowiedzi serwera ma wpływ również czas transmisji danych przez sieć. Jednak w niniejszym opracowaniu uwaga zostanie skupiona jedynie na drugim znaczącym składniku postrzeganego czasu obsługi wynikającego z czasu odpowiedzi samego serwera WWW.

Poza długofalową tendencją wzrostową średniego obciążenia serwerów WWW istnieje także problem szybkozmienności i nieprzewidywalności ruchu, co wymaga odpowiedniego uwzględnienia w konstrukcji funkcji szeregowania zadań. Badania statystyczne wyników pomiarów: wielkości plików, czasów transferu, odstępów czasu pomiędzy kolejnymi żądaniami wykazują, iż wartości te charakteryzują się dużą zmiennością [9]. Dokładniejsza analiza wskazuje, iż wykazują one[1] własność długiego ogona [6] (ang. heavy tail). To negatywne skądinąd zjawisko może być kluczem do zwiększenia postrzeganej jakości usług bez zwiększonych nakładów finansowych na zakup sprzętu o większych mocach. Jako najczęściej przyjmowany rozkład prawdopodobieństwa odwzorowujący tego typu zjawisko jest używany rozkład Pareto [7] ze współczynnikiem rozkładu alpha zbliżonym do 1,0. Jego wartość zależy od konstrukcji danej witryny. Jak podaje np. [10], w badanej próbce współczynnik rozkładu Pareto alpha dla wielkości pobieranego obiektu wynosił od 1,1 do 1,3, natomiast dla odstępów czasu między kolejnymi żądaniami od około 0,6 do 0,9.

Z kształtu rozkładu częstości pobierania przez klientów serwisów WWW obiektów o danej wielkości można wysnuć następujące wnioski:

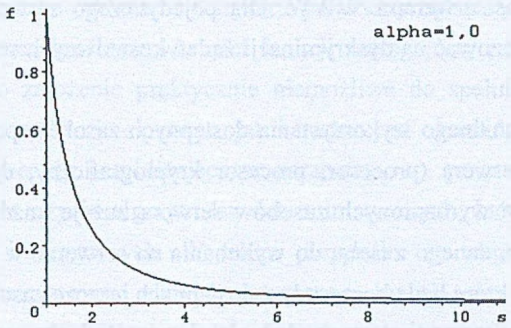
- obsługa większości żądań wymaga użycia niewielkich zasobów[2] i w rezultacie czas wygenerowania odpowiedzi może być krótki,
- w warunkach niemal pełnego lub nadmiernego obciążenia serwera odmowa obsługi żądań najbardziej pracochłonnych[3] pozwoli na obsłużenie wielu innych (ang. heavy tail cutting),
- pojawienie się pracochłonnych żądań może w krótkim czasie spowodować znaczny wzrost obciążenia serwera, a co za tym idzie – opóźnienie w obsłudze żądań mniej pracochłonnych.

[1] Można skonstruować witrynę WWW, dla której np. wielkości pobieranych obiektów nie mają tej własności. Jednak badania nad rzeczywistymi witrynami WWW ją potwierdzają.

[2] Zasobami tymi mogą być: praca dysku, czas utrzymania połączenia, zajętość pasma, wymagana praca CPU (szczególnie przy dynamicznie generowanej treści), używana pamięć i inne zasoby potrzebne do wygenerowania odpowiedzi na żądanie klienta.

[3] Ilość zasobów, które muszą zostać zużyte, aby zwrócić odpowiedź na żądanie klienta.

Zostało praktycznie stwierdzone w wielu pracach (por. [11] [12] [1]), że w warunkach niemal pełnego i nadmiernego obciążenia serwera, do optymalizacji zużycia jego zasobów użyteczne jest priorytetowanie i szeregowanie żądań zgodnie z przyjętym kryterium wynikającym z celu optymalizacji.




Rys. 1. Przykład rozkładu dystrybucji Pareto dla współczynnika alpha=1,0. Interpretacja dla wykresu jako ilustracji częstości pobierania obiektów o danej wielkości: na osi poziomej s jest wielkością obiektu, na osi pionowej f oznacza częstość jego pobierania dla skali, w której najczęściej pobieranej wielkości obiektu przypisano wartość 1,0.

Fig. 1. An example of Pareto distribution with alpha factor = 1.0

## 2. Cechy stosowanych rozwiązań szeregowania zadań

Dla osiągnięcia pojedynczego celu, jakim jest zachowanie wymaganej jakości obsługi, stosowane są rozmaite podejścia. Podstawowe ich cechy przedstawiono w kolejnych podrozdziałach.

### 2.1. Kryterium optymalizacji

Możemy wyróżnić następujące rodzaje kryteriów (pewne z nich mogą być stosowane łącznie):

- wydajnościowe, wśród których występują:
  - minimalizacja czasu obsługi żądania,
  - maksymalne wykorzystanie dostępnych zasobów (procesor, dysk, sieć),
  - nieprzekroczenie maksymalnego czasu obsługi,
- zapewnienia pożądanego QoWS,
- różnorakie biznesowe, np.:

- zapewnienie ciągłości obsługi w ramach sesji,
- minimalizacja czasu obsługi strony,
- uwzględnienie istnienia klas klientów obsługiwanych na różnych zasadach.

Kryterium minimalizacji czasu obsługi jest najczęściej realizowane w systemach z wieloma fizycznymi serwerami WWW. Dla pojedynczego serwera WWW przykładowe rozwiązania muszą bazować na dyskryminacji żądań kosztownych w celu minimalizacji czasu obsługi pozostałych.

Kryterium maksymalnego wykorzystania dostępnych zasobów polega na określeniu krytycznych zasobów serwera (procesor, procesor kryptograficzny, dysk, kanał przesyłowy) i określeniu jak wiele wyróżnionych zasobów serwera zużyje każde z żądań. Zależnie od stopnia wykorzystania danego zasobu do wykonania na serwerze w pierwszym rzędzie kierowane są te żądania, które będą korzystały z dostępnych jeszcze zasobów [12].

Dla kryterium limitowanego czasu obsługi priorytet żądania wzrasta (niekoniecznie liniowo) wraz z upływem czasu, tak aby żądanie zostało wykonane przed upływem przyjętego maksymalnego dopuszczalnego czasu obsługi. Jednym z algorytmów działających według tego kryterium jest EDF (ang. Earliest Deadline First), kolejkujący żądania według czasu pozostałego na ich wykonanie.

Gdy do kryteriów dodane zostanie wymaganie określające, dla jakiej minimalnej części obsługiwanych żądań ma być spełnione kryterium jakości, to otrzymany zostanie system z QoWS.

W rzeczywistych warunkach bardzo rzadko występują pojedyncze żądania dostarczenia pojedynczych obiektów. Stosuje się modyfikacje powyższych algorytmów uwzględniające istnienie sesji klienta, która raz rozpoczęta – powinna się odbywać z należytą jakością obsługi [11]. Podobnie sekwencyjne pobranie pojedynczej strony HTML i wszystkich jej elementów powinno być obsłużone jak najszybciej, co w wyraźny sposób polepsza postrzeganą jakość obsługi[1].

Ponadto z różnych względów może istnieć konieczność rozróżniania poszczególnych klas klientów i zróżnicowania jakości usług im oferowanych.

### 2.2. Podejście probabilistyczne i deterministyczne

Przy zastosowaniu probabilistycznego podejścia zakłada się, że żądania przybywają w nieznanych z góry momentach i że nie są znane czasy ich obsługi. Natomiast znany jest

[1]Optymalizacja pojedynczych żądań może powodować, iż przy nadmiernym obciążeniu serwera zdarzać się będą zerwania sesji w jej trakcie, czy też występować będą inne skutki uboczne, jak np. niemożność pobrania z serwera niektórych elementów właśnie pobranej strony HTML.

rozkład prawdopodobieństwa niosący informację o specyfice obsługiwanego ruchu (choć informacja ta nie musi być wykorzystana). Podstawowym algorytmem należącym do tej klasy jest FIFO (ang. first in – first out), w którym zgłoszenia obsługiwane są zgodnie z kolejnością ich napływania.

Podejście deterministyczne zakłada, iż wszystkie parametry żądań są znane, dzięki czemu możliwe jest jednoznaczne określenie wartości funkcji szeregującej zadania. W przypadku serwerów WWW jest to założenie praktycznie niemożliwe do spełnienia, gdyż zarówno momenty pojawienia się żądań nie są znane, jak i trudno jest określić, np. pracochłonność żądania. Stosowane są metody przybliżające te parametry, np. przez analizę ich historii. Dzięki temu możliwe jest użycie algorytmów deterministycznych, które jednak korzystają z danych przybliżonych i niepewnych. W takim przypadku ze względu na niepewność i niestacjonarność obserwowanych zjawisk użyteczne bywają np. metody sztucznej inteligencji [4]. Dokładniejsze omówienie istniejących metod szeregowania zadań można znaleźć w [3].

### 2.3. Informacje użyte do podjęcia decyzji o szeregowaniu

Możemy wyróżnić następujące informacje użyteczne do podjęcia decyzji o szeregowaniu:

- przewidywanej pracochłonności żądania,
- rejestrowania danych statystycznych na temat rzeczywistych czasów żądań lub ich grup,
- bieżących obciążeń serwera (mierzonych też jako wydajność, np. zadań/sekundę,
- klasyfikacji danego połączenia lub klienta, albo pojedynczego żądania.

Ponieważ przewidzenie dokładnej pracochłonności każdego żądania byłoby praktycznie niemożliwe, to gromadzone są dane na temat czasu obsługi pewnych klas żądań w zależności od obciążenia serwera. Parametr obciążenia serwera może przy tym być rozbity na kilka wielkości opisujących, np. stopień użycia procesora, WE/WY dysku, interfejsu sieciowego czy pamięci[1]. Dane na temat czasów obsługi poszczególnych żądań przy bieżącym obciążeniu mogą być zbierane po wykonaniu każdego żądania albo też jedynie w trybie "uczenia". Do podjęcia decyzji o szeregowaniu żądania bywa również używana informacja o bieżącym obciążeniu serwera.

Używane bywają także informacje pochodzące bezpośrednio z nadchodzących połączeń [8]:

---

[1]Przykładowy podział mógłby obejmować: 1. niewielkie pliki statyczne (małe zużycie wszelkich zasobów), 2. duże pliki (długotrwałe zajęcie pamięci ze względu na dłuższy czas obsługi, zajęcie pasma przesyłowego), 3. treść generowaną dynamicznie (duże zużycie CPU i być może dysku).

- z nagłówków TCP/IP (warstwa 4[1]), z których otrzymuje się dane na temat adresu IP klienta,
- z nagłówków HTTP (warstwa 7), z których otrzymuje się informacje o:
  - sesji klienta (na podstawie cookies),
  - żądanym obiekcie (na podstawie adresu URL).

Informacje te mogą być wykorzystane m.in. do rozpoznawania już rozpoczętych sesji i do rozróżniania klas klientów, którzy mają być obsługiwani na różnych zasadach.

### 2.4. Miejsce i sposób realizacji decyzji o szeregowaniu

Najczęstsze miejsca i sposoby realizacji decyzji o szeregowaniu to:

- aplikacja serwera WWW,
- proxy,
- TCP-SYN (jądro systemu),
- aplikacja uruchomiona na serwerze WWW badająca obciążenie serwera.

Niestety, popularne oprogramowanie serwerów WWW praktycznie nie dostarcza mechanizmów pozwalających realizować własną politykę szeregowania zadań. W przypadku oprogramowania, dla którego źródła są dostępne, często stosowanym rozwiązaniem jest modyfikacja samej aplikacji serwera, w tym przede wszystkim sposobu obsługi kolejki zadań.

Mechanizmem, który może działać praktycznie dla każdego oprogramowania serwera WWW, zwykle stosującego prostą kolejkę FIFO, jest mechanizm proxy. Program typu proxy, działający na tej samej maszynie co oprogramowanie serwera, odpowiada za szeregowanie zadań i przekazywanie ich do wykonania serwerowi.

Wydajnym rozwiązaniem, które stosowane jest jako mechanizm zapobiegający przeciążeniu serwera, jest przejęcie obsługi nawiązywania połączenia TCP/IP przez oprogramowanie inne niż standardowa procedura zawarta w jądrze systemu. Dzięki opóźnieniu odpowiedzi na pakiet nawiązujący połączenie (TCP-SYN) lub wręcz odesłaniu odpowiedzi odmowy połączenia (RST) możliwa jest skuteczna i wydajna kontrola ilości przyjmowanych przez serwer połączeń (ang. admission control) w warunkach przeciążenia. Niestety, metoda ta wymaga ingerencji w jądro systemu.

Prymitywna, choć skuteczna kontrola obciążenia systemu, może być wykonywana przez skrypty CGI. Jeśli obciążenie systemu jest większe od dopuszczalnego, to skrypt nie wykonuje pracochłonnego zadania, a jedynie zwraca klientowi informację o chwilowym przeciążeniu systemu.

---

[1] Modelu warstwowego protokołów sieciowych ISO.

## 3. Przykładowe implementacje szeregowania zadań w serwerze WWW

Na przykładzie pracy [1] zostaną wskazane charakterystyczne cechy rozwiązania:

- kryteria optymalizacji: nieprzekroczenie maksymalnego czasu obsługi przy maksymalizacji wykorzystania zasobów serwera (tylko ze względu na użycie procesora),
- zastosowane podejście – probabilistyczne (kolejka FIFO dla nieodrzuconych żądań),
- informacje użyte do podjęcia decyzji: obciążenie serwera i klasyfikacja klienta,
- miejsce i sposób realizacji decyzji: aplikacja serwera WWW, odrzucanie żądań.

Na podstawie [2] autorzy przyjęli, iż aby obsłużyć zadania posiadające indywidualne maksymalne czasy obsługi i uszeregowane wedle tych czasów zgodnie z kryterium EDF (ang. Earliest Deadline First) wystarczające jest utrzymanie obciążenia serwera (ang. utilization) poniżej granicy $U < ln\ 2$. Ponieważ przyjęto stały maksymalny czas wykonania każdego z zadań – kryterium EDF mogło zostać zrealizowane za pomocą kolejki FIFO, do której dopuszczano jedynie wybrane żądania, a pozostałe odrzucano. Przy takiej konstrukcji mechanizmu szeregowania i założeniu obsługi głównie statycznych elementów stron WWW $U$ ma postać:

$$U = aRf + bW + cR(1 - f),$$

gdzie:

$R$ – szybkość obsługi żądań,

$W$ – ilość danych zwróconych przez obsłużone żądania,

$f$ – ułamek żądań dopuszczonych do wykonania,

$1 - f$ – ułamek żądań odrzuconych,

$a$, $b$ i $c$ – stałe zależne od systemu, na którym działa serwer WWW, parametryzujące: koszt stały przyjęcia pojedynczego żądania, koszt zależny od wielkości żądania i koszt odrzucenia żądania.

Decyzja o odrzuceniu żądania była podejmowana na podstawie danych o kliencie, na podstawie zdefiniowanych klas klientów o różnych priorytetach. Przeprowadzone eksperymenty potwierdziły skuteczność przyjętego mechanizmu szeregowania. Ze względów wydajnościowych do odrzucania połączeń zalecane jest użycie mechanizmu TCP-SYN.

W pracy [12] przedstawiono nieco inne rozwiązanie. Oto jego cechy:

- kryterium optymalizacji: maksymalizacja wykorzystania zasobów serwera w rozbiciu na zasoby i unikanie jego przeciążenia (czyli kryterium maksymalnego czasu obsługi – por. [1]),
- zastosowane podejście: probabilistyczne z użyciem informacji o specyfice ruchu,

- informacje użyte do podjęcia decyzji: przewidywana pracochłonność żądania i obciążenie serwera w rozbiciu na zasoby, klasyfikacja IP klienta, miejsce i sposób podjęcia decyzji: kontrola przyjęć TCP-SYN i 2 kolejki – w jądrze.

Autorzy przyjęli, iż istnieją dwa krytyczne zasoby serwera WWW: procesor i przepustowość sieci. Mając to na uwadze, elementy witryny podzielono na trzy grupy:

- obciążające procesor (skrypty CGI),
- zajmujące kanał przesyłowy,
- pozostałe.

Przychodzące połączenia były odrzucane w przypadku przeciążenia serwera na podobnych zasadach jak w [1], z użyciem mechanizmu TCP-SYN. Przepuszczone dalej połączenia były nawiązywane, a na podstawie przesłanego przez klienta żądania – klasyfikowane do jednego z dwóch token bucketów lub w przypadku trzeciej grupy – przepuszczane bezpośrednio do niezmodyfikowanego serwera z kolejką FIFO. Wielkość token bucketów była odpowiednio regulowana w zależności od bieżącego obciążania procesora i kanału przesyłowego.

Ilość żądań obciążających procesor *Rcpu* przyjmowanych w przedziale czasu $t + 1$ wynosiła:

$$Rcgi(t+1) = Rcgi(t) + Kcpu \cdot (CPUref - CPUcurr),$$

gdzie:

*Rcgi* – ilość żądań przyjmowanych w przedziale czasu,

*Kcpu* – parametr decydujący o szybkości zmiany wielkości token bucket,

*CPUref* i *CPUcurr* – odpowiednio: pożądane i bieżące obciążenie CPU.

Analogiczny sposób sterowania długością kolejki przyjęto dla kolejki żądań zajmujących kanał przesyłowy i dla regulacji odrzucania żądań za pomocą mechanizmu TCP-SYN. Przeprowadzone eksperymenty dowiodły, iż zastosowanie tego mechanizmu pozwoliło zapewnić znacząco krótszy średni czas odpowiedzi w pełni obciążonego i przeciążonego serwera niż bez jego użycia.

## 4. Dalsze kierunki rozwoju badań

Przedstawione tutaj dwa dość proste przykłady mechanizmów szeregowania zadań nie oddają bogactwa istniejących rozwiązań, częstokroć znacznie bardziej komplikowanych. Wydaje się jednak, iż pewne cechy rozwiązań występują częściej niż inne. Dla przykładu większość rozwiązań charakteryzuje podejście probabilistyczne ze względu na nieprzewidywalność czasów nadejścia żądań i ich parametrów (czasochłonności, czasu wykonania). Istnieją jednak techniki z zakresu sztucznej inteligencji, które mogą posłużyć jako mechanizmy

dostarczania danych niepewnych na temat nadchodzących żądań, które następnie mogą być użyte w standardowych deterministycznych algorytmach szeregowania.

Kluczem umożliwiającym prawidłowe użycie tych uznanych i dobrze opisanych algorytmów jest jednak zapewnienie im danych wejściowych o odpowiedniej jakości. Zagadnienia z tym związane będą rozpatrywane szerzej w ramach dalszych prac.

## LITERATURA


1. Abdelzaher T. F., Shin Kang G.: Performance guarantees for web server end-systems: a control-theoretical aproach. Transactions on Parallel and Distributed Systems, Vol. 13, No. 1, Jan 2002.
2. Audsley N. C.: Deadline monotonic scheduling theory. 18th IFAC Workshop on Real Time Programming, Belgium 1992.
3. Błażewicz J., Cellary W., Słowiński R., Węglarz J.: Badania operacyjne dla informatyków, WNT, Warszawa 1983.
4. Borzemski L., Zatwarnicki K.: A fuzzy adaptive algorithm for cluster based web systems. IEEE Computer Society 2003.
5. Colajanni M., Yu Philip S., Cardellini V.: Scalable Web Server Systems: archtectures, models and load-balancing algorithms, Sigmetrics 2000.
6. Crovella E. M., Bestavros A.: Explaining World Wide Web Traffic Self-Similarity, Technical Report TR-95-015, 1995.
7. Feitelson D. G.: Random Number Generators and Heavy-Tail Distributions. Technical Report 2001-2, School of Computer Science and Engineering, The Hebrew University of Jerusalem.
8. Feldmann A.: BLT: Bi-layer tracing of HTTP and TCP/IP, AT&T Labs-Research, NJ,USA. 9th International World Wide Web Conference, Amsterdam 2000,
9. Feldmann A., Whitt W.: Fitting mixtures of expotentials to Long-tail distributions to analyze network performance models". IEEE INFOCOM 1997, ss. 1098-1116.
10. Hairong Sun: The effect of web caching on network planning. Computer Communications, Vol. 22, No.14, pp. 1343-1350, Sept. 1999, Elsevier Science
11. Huamin Chen, Mohapatra P.: Session-based overload control in QoS-aware web servers. IEEE INFOCOM 2002, New York, Jun. 2002.
12. Thiemo Voigt, Per Gunningberg: Handling Multiple Bottlenecks in Web Servers Using Adaptive Inbound Controls International Workshop on Protocols For High-Speed Networks, April 2002, Berlin, Germany.

Recenzent: Dr inż. Wojciech Mikanik

Wpłynęło do Redakcji 8 kwietnia 2003 r.

**Abstract**

This paper's focus are different goals and thus different features of existing and possible web server task scheduling mechanisms. They are indentified in the following areas: optimalization criteria, usage of probabilistic or deterministic approach, informations used for scheduling decision, place and method of decision realization. As an example – this paper points out features of two choosen scheduling solutions. Basing on wider literature studies it proposes some apparently interesting, not yet commonly explored research areas.

**Adres**

Grzegorz B. PROKOPSKI: Politechnika Wrocławska, Instytut Sterowania i Techniki Systemów, ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Polska, gadek@debian.org .