ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: INFORMATYKA z. 30 Nr kol. 1315

Tadeusz CZACHÓRSKI
Michał PASTUSZKA

EFEKTYWNOŚĆ RÓWNOLEGŁEGO WYKONANIA ZADAŃ W SYSTEMACH ROZPROSZONYCH; MODELE TYPU FORK-JOIN

Streszczenie. Artykuł ocenia efektywność równoległego wykonania zadań w rozproszonych systemach komputerowych, posługując się w tej ocenie zaczerpniętym z teorii kolejek modelem *fork-join* oraz pokrewnymi mu modelami *fission-fusion* i *split-merge*. Rezultaty wskazują, że przyspieszenia wynikające z równoległej organizacji pracy są w istotny sposób zmniejszone przez ograniczenia wnoszone przez synchronizację zadań.

EFFICIENCY OF PARALLEL EXECUTION IN DISTRIBUTED SYSTEMS: FORK-JOIN MODELS

Summary. *Fork-join, split-merge* and *fission-fusion* models are a suitable tool to evaluate the efficiency of execution of jobs defined by parallel-sequential task precedence graph in multiprocessor environment. The results indicate that the speedup which can be expected theoretically in a parallel system may be decreased significantly by synchronization constraints.

EFFICACITE D'EXECUTION PARALLELE DANS LES SYSTEMES DISTRIBUES: MODELES DE TYPE FORK-JOIN

Résumé. Les modèles *fork-join, split-merge* et *fission-fusion* présentent une outil commode pour étudier l'efficacité d'éxécution des programmes decrits par un graph parallèle-séquetiele de précédence dans un milieu multiprocesseur. Les résultats indiquent que la pérformance q'on peut théoriquement prévoir dans un système parallèle est gravement diminuée par les contraints de synchronisation.

Niniejszy tekst został opracowany w ramach Projektu Badawczego KBN nr 8 T11C 032 08.

1. Wstęp

Strukturę programu (lub innego wykonywanego zadania) można przedstawić grafem szeregowo-równoległym, opisującym kolejność wykonania fragmentów programu: wykonywane części są węzłami grafu, a łuki określają następstwo podzadań. Przykład takiego grafu przedstawia rys. 1. Zwraca on uwagę na fakt, że w systemie złożonym z wielu procesorów lub komputerów, nawet przy braku ograniczeń co do liczby wykorzystywanych procesorów i co do objętości pamięci oraz zakładając brak konfliktów przy dostępie do niej, możliwości równoległego wykonania programu są z reguły ograniczone przez jego strukturę, w której obok elementów niezależnych, które mogą być wykonane równolegle, występują elementy, które muszą być wykonane sekwencyjnie. Podstawowym elementem takiego grafu jest struktura przedstawiona na rys. 2: operacje $O_1, O_2, \ldots, O_L$ mogą być wykonywane równolegle, ale dopiero po zakończeniu operacji O_0; warunkiem rozpoczęcia operacji O_{L+1} jest zakończenie wszystkich operacji $O_1, \ldots, O_L$.

W przypadku idealnym, gdy proces nigdy nie czeka na przydział zasobu, czasowi wykonania zadania odpowiada czas przejścia przez graf o zadanej strukturze szeregowo-równoległej, przy czym z każdym węzłem grafu związany jest czas pobytu — zmienna losowa o zadanym rozkładzie. Przypadek taki analizują m.in. prace [12, 4].

W artykule zajmujemy się innym przypadkiem, który opisuje wykonanie zadań w systemie rozproszonym. Zakładamy, że każda z operacji wyszczególnionych w grafie może być wykonana przez oddzielny procesor, który jednakże wykonuje też inne zadania, a więc występują przed nim kolejki. Wykonanie programu, którego operacje są uszeregowane jak na rys. 2, przekładamy na model kolejkowy jak na rys. 3a, zwany modelem *fork-join*.

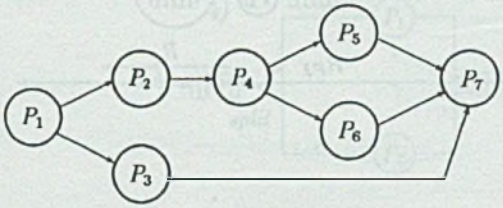
Zakładamy, że klient dochodząc do punktu *fork*, rozpada się na L nowych klientów (zadań cząstkowych), obsługiwanych równolegle w L stanowiskach $P_1, P_2, \ldots, P_L$. Struktura ta odpowiada wykonaniu części programu zapisanej właśnie za pomocą konstrukcji fork-join lub parbegin-parend [8]. W każdym ze stanowisk mogą być zadania cząstkowe pochodzące z rozpadu poprzednich zadań, a więc mogą być kolejki różnej długości. Za równoległymi stanowiskami znajduje się poczekalnia, zwana *kolejką join* lub *kolejką synchronizacyjną*, mieszcząca wszystkich klientów, którzy zakończyli już obsługę i czekają na tych swoich współbraci; których obsługa jeszcze się nie zakończyła (lub, być może, jeszcze nie zaczęła, bo czekają oni w kolejce do przydzielonego im stanowiska). Po skompletowaniu wszystkich L zadań pochodzących z rozszczepienia tego samego klienta łączą się one w jedność — z kolejki join ubywa L klientów, odpowiadających wykonanym zadaniom cząstkowym, a do kolejki następnego stanowiska przechodzi jeden klient.

Podział zadania na wykonywane równolegle części skraca czas jego realizacji — z tego punktu widzenia im wyższy stopień równoległości, tym krótszy czas realizacji. Ponieważ jednak istnieje kolejka synchronizacyjna, zatem im więcej partnerów, na których ukończenie obsługi zadanie musi czekać, by opuścić tę kolejkę, tym dłuższy czas poświęcony późniejszej synchronizacji elementów zadania. Wprowadźmy następujące parametry wyrażające te zależności.

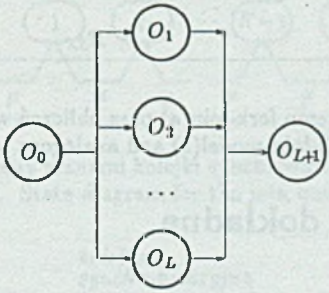
- *Przyspieszenie* G — uśredniony czas sekwencyjnego wykonania wszystkich L części zadania (czas R_{SZ} na rys. 3b) podzielony przez średni czas ich równoległego wykonania w systemie fork-join, z uwzględnieniem czasu synchronizacji (czas R_{FJ} na

rys. 3b), $G = \overline{R}_{SZ}/\overline{R}_{FJ}$. Kreska nad symbolem zmiennej oznacza wartość średnią.

- *Narzut synchronizacyjny* S — stosunek średniego czasu spędzonego w kolejce synchronizacyjnej do całego czasu spędzonego w strukturze fork-join: $S = \overline{R}_J/\overline{R}_{FJ}$.
- *Współczynnik blokowania* B — średnia liczba zadań cząstkowych znajdujących się w kolejce synchronizacyjnej.



Rys. 1. Przykład grafu szeregowo-równoległej realizacji zadania
Fig. 1. Exemplary parallel-series task precedence graph

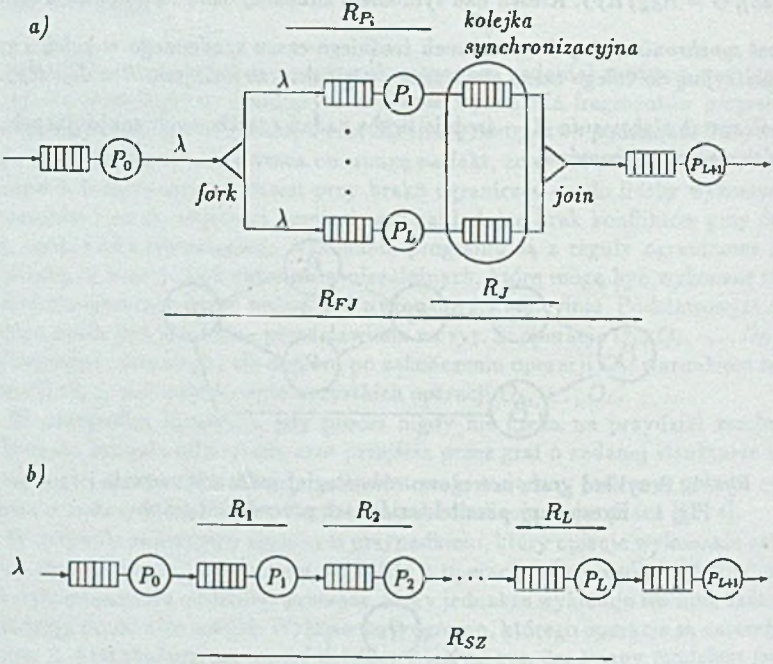


Rys. 2. Graf następstwa zadań, którego model kolejkowy przedstawiony jest na rys. 3a
Fig. 2. Task precedence graph corresponding to queueing network in Fig. 3a

W przypadku struktury *fission-fusion* (rys. 4a) L *dowolnych* zadań cząstkowych może połączyć się ze sobą. Kolejka synchronizacyjna opróżnia się natychmiast, gdy znajdzie się w niej L podzadań, nawet jeśli pochodzą one z podziału różnych zadań.

W strukturze *split-merge* (rys. 4b) nie ma kolejek do równoległych procesorów; zadania czekają na podział w kolejce przed punktem *split* i dopiero, gdy zakończy się wykonanie wszystkich podzadań danego zadania, następne może się podzielić.

W dalszych rozdziałach omówimy stosowane w literaturze modele tych struktur (najwięcej rezultatów dotyczy struktury fork-join). Każda z tych metod ma swoje ograniczenia, dlatego w ostatnim rozdziale wykorzystano metodę symulacji zdarzeń dyskretnych w celu określenia parametrów pracy omawianych struktur.

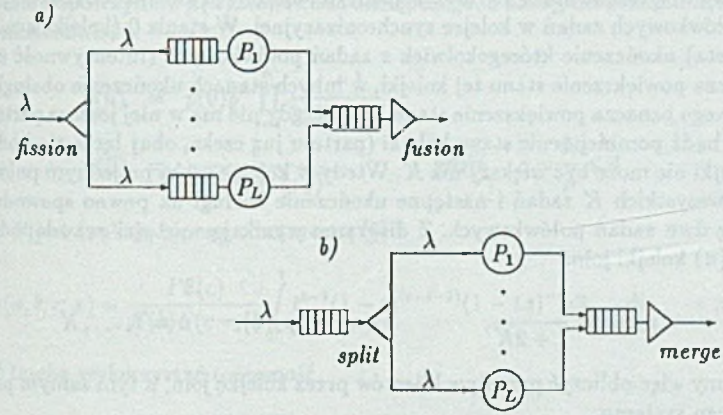


Rys. 3. Model systemu fork-join(a) oraz obliczeń wykonanych szeregowo(b)
Fig. 3. Fork-join model(a) and analogous sequatial system(b)

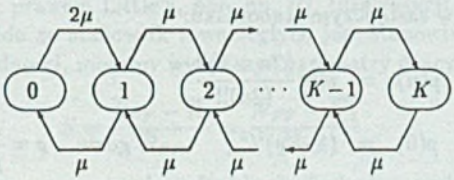
2. Rezultaty dokładne

2.1. Modele analityczne

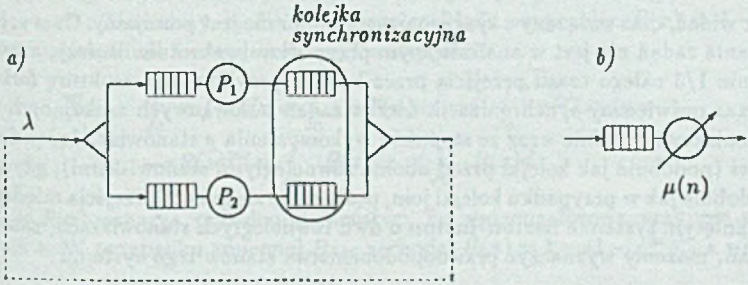
Rozpocznijmy analizę systemu **fork-join** od prostego przypadku. Załóżmy, że zadania przychodzą w odstępach o rozkładzie wykładniczym ze stałym parametrem λ, że każde zadanie dzieli się na dwie wykonywane równolegle części ($L = 2$) i że obsługa obu części ma rozkład wykładniczy ze stałym parametrem μ. W sieciach markowskich można układ wielu stanowisk zastąpić jednym stanowiskiem o odpowiednio dobranym parametrze $\mu(n)$ takim, by przepustowość stanowiska zastępczego była zawsze taka sama, jak przepustowość analizowanego zespołu stanowisk. Zgodnie z tą regułą system fork-join na rys. 6a zastąpimy stanowiskiem jak na rys. 6b. Przepustowość systemu fork-join w funkcji liczby obecnych w nim klientów obliczamy zwierając (linia przerywana na rys. 6a) jego wyjście i wejście: klienci, natychmiast po połączeniu w kolejce join, przechodzą do punktu fork i są rozszczepiani. Zastanówmy się nad rozkładem liczby klientów w kolejce join tego systemu. Na rys. 5 jest przedstawiony diagram przejść między stanami kolejki dla przy-



Rys. 4. Systemy fission-fusion (a) i split-merge (b)
Fig. 4. Fission-fusion and split-merge models



Rys. 5. Diagram przejść między stanami kolejki synchronizacyjnej systemu fork-join
Fig. 5. State diagram for the join queue



Rys. 6. Najprostszy model fork-join i równoważne mu stanowisko ze zmiennym parametrem $\mu(n)$

Fig. 6. The simplest non-trivial fork-join model and its equivalent server with state-dependent service rate

padku, gdy w systemie krąży K rozszczepianych i łączonych klientów. Stan jest określony liczbą połówkowych zadań w kolejce synchronizacyjnej. W stanie 0 (kolejka synchronizacyjna pusta) ukończenie któregokolwiek z zadań połówkowych (intensywność zdarzenia 2μ) oznacza powiększenie stanu tej kolejki, w innych stanach ukończenie obsługi zadania połówkowego oznacza powiększenie stanu kolejki (gdy nie ma w niej jeszcze partnera tego zadania) bądź pomniejszenie stanu kolejki (partner już czeka, obaj łączą się i odchodzą). Stan kolejki nie może być większy niż K. Wtedy w kolejce czeka po jednym połówkowym zadaniu wszystkich K zadań i następne ukończenie obsługi na pewno spowoduje połączenie się dwu zadań połówkowych. Z diagramu wynikają wartości prawdopodobieństw stanu $p_J(n)$ kolejki join:

$$p_J(0) = \frac{1}{1+2K}, \qquad p_J(n) = 2p_J(0), \qquad n = 1, \ldots, K \tag{1}$$

Możemy więc obliczyć przepływ klientów przez kolejkę join, a tym samym przepustowość całego systemu

$$d(K) = \mu \sum_{i=1}^{K} p_J(i) = \frac{2K}{1+2K}\mu \tag{2}$$

i dobrać współczynnik $\mu(n) \equiv d(n)$ dla stanowiska zastępczego. Można też obliczyć rozkład liczby klientów w zastępczym stanowisku:

$$p(n) = p(0)\frac{(2n+1)!!}{(2n)!!}, \tag{3}$$

$$p(0) = (1-\varrho)^{2/3}, \qquad \text{gdzie} \quad \varrho = \frac{\lambda}{\mu} \tag{4}$$

Obliczamy współczynniki:

$$G = \frac{4}{3}, \qquad S = \frac{1}{3}, \qquad B = \frac{1}{2}\frac{\varrho}{1-\varrho}$$

Jak widać, czas związany z synchronizacją zadań nie jest pomijalny. Czas szeregowego wykonania zadań nie jest w analizowanym przypadku dwukrotnie dłuższy, a tylko o 1/3. Podobnie 1/3 całego czasu przejścia przez badaną przykładową strukturę fork-join stanowi czas poświęcony synchronizacji. Liczba zadań połówkowych czekających w kolejce synchronizacyjnej rośnie wraz ze stopniem wykorzystania ϱ stanowisk, dążąc do nieskończoności (podobnie jak kolejki przed oboma równoległymi stanowiskami), gdy $\varrho \to 1$.

Podobnie jak w przypadku kolejki join, posługując się grafem przejścia między stanami w zamkniętym systemie **fission-fusion** o dwu równoległych stanowiskach, zawierającym K zadań, możemy wyznaczyć prawdopodobieństwa stanów tego systemu

$$p_{FF}(0) = \frac{1}{4K-1}, \qquad p_{FF}(n) = 2p_{FF}(0), \qquad n = 1, \ldots, K \tag{5}$$

jego przepustowość, gdy jest zwarty

$$d(K) = \frac{2K}{1+2K}\mu, \tag{6}$$

która określa współczynnik $\mu(n)$ stanowiska zastępczego, a następnie rozkład liczby klientów w tym stanowisku:

$$p(n) = p(0)\varrho^n \prod_{i=1}^{K} \frac{4i-1}{2(2i-1)}, \tag{7}$$

$$p(0) = \frac{1}{{}_2F_1(1, \frac{3}{4}; \frac{1}{2}; \varrho)}, \qquad \text{gdzie} \quad \varrho = \frac{\lambda}{\mu} \leqslant 1, \tag{8}$$

a funkcja F jest funkcją hipergeometryczną [3]

$${}_2F_1(a,b;c;x) = \frac{\Gamma 2(c)}{\Gamma(b)\Gamma(c-b)} \int_0^1 t^{b-1}(1-t)^{(c-b-1)}(1-tx)^{-a}dt \qquad \text{dla} \qquad c > b;$$

dla $c \leqslant b$ trzeba wykorzystać tożsamość

$${}_2F_1(a,b;c;x) = (1-x)^{c-a-b}\,{}_2F_1(c-a, c-b; c; x);$$

$\Gamma(x)$ jest funkcją gamma.

Rozkład $p(n)$ pozwala wyznaczyć wartość średnią $\overline{N}_{FF}$ liczby zadań w całym systemie, a także, posługując się prawem Little'a, por. np. [7], czas reakcji $\overline{R}_{FF} = \overline{N}_{FF}/\lambda$ tego systemu. Ponieważ każde ze stanowisk równoległych jest stanowiskiem $M/M/1$ i jego analiza nie stwarza trudności, możemy wyznaczyć parametry pracy całego systemu, np.

$$S = \frac{\overline{R}_{FF} - \overline{R}_1}{\overline{R}_{FF}} = \frac{\overline{N}_{FF} - \overline{N}_1}{\overline{N}_{FF}}$$

Stanowisko **split-merge**, zawierające L równoległych stanowisk, widziane w całości, zachowuje się dokładnie tak, jak pojedyncze stanowisko $M/G/1$, w którym czas obsługi jest równy najdłuższemu z czasów obsługi w równoległych stanowiskach: $B_{SM} = \max\{B_1, B_2, \ldots, B_L\}$. Zakładamy, że rozkład czasów obsługi jest we wszystkich stanowiskach wykładniczy z tym samym parametrem μ, czyli dystrybuantą tego rozkładu jest $F(x) = 1 - e^{-\mu x}$. Zauważmy, że dla dowolnego rozkładu oznaczając $R(x) = 1 - F(x)$, możemy wyrazić wartość średnią zmiennej losowej jako:

$$\overline{X} = \int_0^\infty x f(x)dx = \int_0^\infty x F'(x)dx = -\int_0^\infty x R'(x)dx =$$
$$-xR(x)|_0^\infty + \int_0^\infty R(x)dx = \int_0^\infty R(x)dx$$

Funkcja $R(x)$ oznacza prawdopodobieństwo, że zmienna losowa przyjmie wartość większą niż x. W przypadku zmiennej B_{SM} zachodzi $R(x) = 1 - (1 - e^{\mu x})^L$, a więc

$$\overline{B}_{SM} = \int_0^\infty \left(1 - (1 - e^{\mu x})^L\right) dx$$

Po wprowadzeniu zmiennej $u = 1 - (1 - e^{\mu x})$, skąd $dx = \frac{du}{\mu e^{-\mu x}} = \frac{du}{\mu(1-u)}$ obliczamy:

$$\overline{B}_{SM} = \frac{1}{\mu}\int_0^1 \frac{1-u^n}{1-u}du = \frac{1}{\mu}\int_0^1 \left(\sum_{i=1}^{L} u^{i-1}\right) du = \frac{1}{\mu}\sum_{i=1}^{L}\int_0^1 u^{i-1}du = \frac{1}{\mu}\sum_{i=1}^{L}\frac{1}{i}, \tag{9}$$

zatem B_{SM} ma średnią $\frac{1}{\mu}H_L$, gdzie $H_L = \sum_{i=1}^{L}\frac{1}{i}$; podobnie można obliczyć jej wariancję:

$$Var[B_{SM}] = \frac{1}{\mu^2}\sum_{i=1}^{L}\frac{1}{i^2} = \frac{1}{\mu^2}H_L^{(2)}, \tag{10}$$

gdzie $H_L^{(2)} = \sum_{i=1}^{L}\frac{1}{i^2}$, por. [11]. Pozwala to, posługując się wzorem Pollaczka-Chinczyna dla stanowiska $M/G/1$ por. np. [7], określić średnią liczbę zadań (z których jedno jest podzielone na części) w systemie split-merge:

$$\overline{N}_{SM} = \varrho + \frac{\lambda^2\overline{B^2}}{2(1-\varrho)} = \varrho + \frac{\varrho^2(H_L^2 + H_L^{(2)})}{2(1-\varrho)}, \tag{11}$$

a następnie, w podobny jak poprzednio sposób, miary pracy systemu.

2.2. Obliczenia numeryczne

Można konstruować ogólniejsze niż przedstawione dotąd modele markowskie o dowolnej liczbie równoległych kanałów, z rozkładem czasu obsługi złożonym z faz o rozkładzie wykładniczym (jest to warunek pozostania w sferze modeli Markowa) tak dobranych, że ich zespół imituje dowolny rozkład. To samo odnosi się do rozkładów odstępów czasu między zgłoszeniami. Szczególnie przydatny jest tu rozkład Coxa: zmienna losowa o rozkładzie Coxa rzędu k ($k \geqslant 1$, całkowite) jest to czas przejścia przez sieć faz. Wszystkie fazy mają rozkłady wykładnicze o parametrach $\mu_1, \ldots, \mu_k$. Faza pierwsza istnieje na pewno, faza $i+1$ następuje po fazie i, ($i \geqslant 1$), z prawdopodobieństwem a_i lub (z prawdopodobieństwem $1-a_i$) jest fazą ostatnią; $a_k = 0$. Za pomocą rozkładu Coxa można aproksymować z dowolną dokładnością każdy rozkład $F_X(x)$ nieujemnej zmiennej losowej o skończonej wartości średniej, por. [10]: dla dowolnego $\varepsilon > 0$ istnieje taki rozkład Coxa $F_C(x)$, że

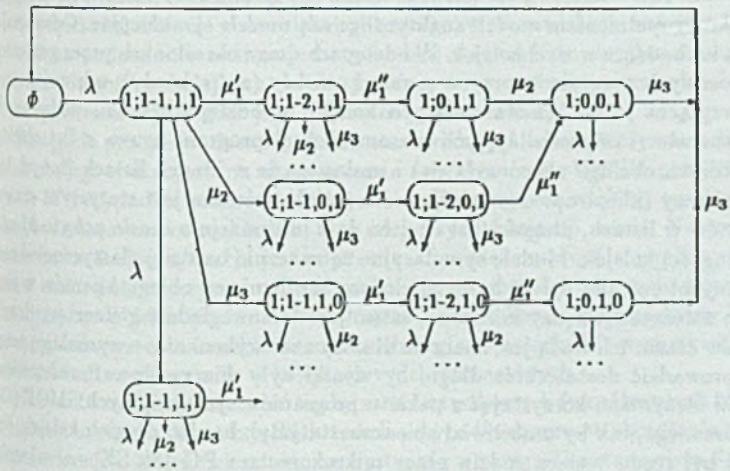
$$\int_0^\infty \left|F_X(x) - F_C(x)\right| dx < \varepsilon \qquad \text{oraz} \qquad \left|\int_0^\infty x[F_X(x) - F_C(x)]dx\right| < \varepsilon,$$

nie ma jednak ogólnej zasady, jak dobrać stopień rozkładu i jego współczynniki.

Jeżeli czas obsługi ma rozkład Coxa, to stan stanowiska charakteryzuje nie tylko — jak to miało miejsce w stanowisku o wykładniczym rozkładzie czasu obsługi — liczba zadań w stanowisku, lecz także faza obsługi zadania aktualnie wykonywanego. Możemy tworzyć diagramy przejść między stanami stanowiska oraz pisać odpowiadające im równania łączące prawdopodobieństwa stanów, trzeba jednak powstałe równania rozwiązać numerycznie.

Rysunek 7 przedstawia fragment diagramu przejść między stanami modelu fork-join posiadającego trzy równoległe stacje obsługi. Czas obsługi w pierwszej z tych stacji charakteryzuje się rozkładem Coxa drugiego rzędu (parametry kolejnych faz tego rozkładu to μ_1' i μ_1''); parametry wykładniczych rozkładów w drugiej i trzeciej stacji to μ_2 i μ_3. Parametr poissonowskiego strumienia wejściowego ma wartość λ. Stan systemu charakteryzuje wektor $(n, n_1\text{-}f, n_2, n_3)$, gdzie n jest liczbą zadań w modelu, n_1 to liczba zadań cząstkowych w pierwszym stanowisku, f jest fazą rozkładu Coxa, w której znajduje się obsługa w tym stanowisku; n_2, n_3 są liczbami zadań cząstkowych w drugim i trzecim stanowisku.

Podstawowym ograniczeniem tej metody jest bardzo szybki wzrost liczby stanów wraz z rozbudową modelu, a więc wzrost liczby równań, które należy rozwiązać. Liczba ta w prostych modelach sięga dziesiątków tysięcy. Wraz ze wzrostem mocy obliczeniowej dostępnych stacji roboczych, z rozwojem potrzebnego oprogramowania (programy automatycznie tworzące na podstawie opisu modelu macierz prawdopodobieństw przejść) oraz specjalnie rozwijanych metod numerycznych rozwiązywania bardzo dużych układów równań o rzadkich macierzach (np. metoda Arnoldiego), por. [6], podejście numeryczne nabiera praktycznego znaczenia.



Rys. 7. Fragment diagramu przejść między stanami modelu fork-join, zawierającego 3 równoległe stanowiska, z których pierwsze ma rozkład obsługi Coxa 2 rzędu

Fig. 7. A fragment of state diagram of fork-join model with 3 parallel servers; the first of them has Cox-2 service time distribution

Algorytm rozwiązywania sieci stanowisk zawierających elementy fork-join, pozwalający zmniejszyć rozpatrywaną przestrzeń stanów, podaje praca [8]. Najpierw stanowiska fork-join są traktowane osobno — są one zwierane i oblicza się numerycznie współczynniki obsługi $\mu(n)$ dla równoważnych im stanowisk, później rozwiązywana jest numerycznie sieć ze stanowiskami równoważnymi.

2.3. Modele przybliżone

Modele przybliżone, których nie będziemy tu omawiać, starają się rozwiązać model struktury fork-join różnymi sposobami:

- poprzez podanie górnej i dolnej granicy zachowania się elementu fork-join, zastępując go odpowiednio dobranymi stanowiskami $D/G/1$ oraz $G/G/1$: modele Baccellego i Makowskiego, m.in. [2],

- poprzez adaptację analizy wartości średnich *MVA* [1],
- wykorzystanie aproksymacji dyfuzyjnej [5].

3. Badania symulacyjne

Naturalnym uzupełnieniem modeli analitycznych są *modele symulacyjne*. Odtwarzają one zdarzenia zachodzące w sieci kolejek. W odstępach czasu określonych przez generator liczb pseudolosowych program tworzy w pamięci obiekty (np. rekordy), odpowiadające klientom i przyłącza je do list stanowiących kolejki; w odstępach czasu podawanych przez inny generator i odpowiadających czasom obsługi program usuwa z list obiekty (to klienci, których obsługa zakończyła się) i umieszcza je w innych listach (innych kolejkach) lub niszczy (klient opuszcza sieć). Jednocześnie zbierana jest statystyka czasów pobytu obiektów w listach, długości list itp., co daje informacje o czasie pobytu klienta w kolejce i długości kolejek. Modele symulacyjne są znacznie bardziej elastyczne od modeli analitycznych: potrafią opisać bardziej złożone mechanizmy obiegu klientów w sieci i wzajemnych zależności między klientami, łatwo jest też uwzględnić generatory dowolnych rozkładów czasu. Ich wadą jest znacznie dłuższy czas wykonania — symulację pracy sieci trzeba prowadzić dostatecznie długo, by wyniki były wiarygodne. Przedstawione poniżej wyniki otrzymano korzystając z pakietu programów symulacyjnych SMOK [13], nieco przetworzonego, tak by umożliwiał zbieranie statystyk bardzo długich kolejek. Nakład obliczeń był rzędu tysiąca godzin pracy mikrokomutera PC 486 SX, co wskazuje, jak żmudne jest wykorzystanie modeli symulacyjnych w przypadku analizy wielu wariantów modelu. Wybrane wyniki przedstawiono na rys. 8–11. Prezentują one kolejno zależność współczynnika synchronizacji (procent czasu spędzonego w systemie poświęconego synchronizacji zadań) oraz współczynnika blokowania (długość kolejek synchronizacyjnych) w stanowiskach fork-join, fission-fusion i split-merge od stopnia obciążenia systemu oraz od współczynnika zmienności rozkładu czasu obsługi w równoległych stanowiskach. Współczynnik zmienności rozkładu jest definiowany jako iloraz wariancji przez kwadrat wartości średniej; dla rozkładu wykładniczego równa się on jedności. Reprezentantem rozkładów niewykładniczych był rozkład Coxa drugiego rzędu, którego parametry były obliczane na podstawie ustalonej wartości średniej $1/\mu$ i współczynnika zmienności C_X^2 jako: $\mu_1 = 2\Theta\mu$, $\mu_2 = 2(1-\Theta)\mu$, $a = 1 - \Theta - \frac{(1-\Theta)^2}{\Theta}$, gdzie $\Theta = 0.5\left(1 + \sqrt{\frac{C_X^2-1}{C_X^2+1}}\right)$ i dobrane tak, by współczynnik zmienności wynosił 2.5 lub 10. Przypadek zaznaczony na wykresach jako Exp&Cox to model, w którym w jednym z równoległych stanowisk jest rozkład Coxa o współczynniku zmienności wynoszącym 10, a we wszystkich pozostałych stanowiskach są rozkłady wykładnicze. Jak widać na rys. 8, współczynnik synchronizacji w modelu fork-join rośnie wraz ze wzrostem liczby równoległych stanowisk. Jest to naturalne; im więcej zgłaszających się losowo partnerów musi się połączyć, tym dłużej to trwa. Ten sam współczynnik maleje ze wzrostem obciążenia systemu – długość kolejki synchronizacyjnej oczywiście rośnie, co można zobaczyć na rys. 10, lecz wolniej niż długość kolejek przed równoległymi stacjami obsługi. Spadek wartości współczynnika synchroni-

zacji wraz ze wzrostem wykorzystania stanowisk jest szczególnie widoczny w stanowisku fission-fusion, w kolejce synchronizacyjnej którego łączą się dowolne podzadania.

Rysunki 8-9 przedstawiają udział procentowy czasu synchronizacji w całkowitym przejściu zadania przez stanowisko. W przypadku fork-join (rys.3a) i fission-fusion (rys.4a) czas synchronizacji w modelu symulacyjnym był mierzony w kolejce synchronizacyjnej. W stanowisku typu split-merge (rys.4b) jest to łączny czas spędzony przez zadanie w kolejce przed stanowiskiem split i w kolejce synchronizacyjnej. Kolejka *split* rośnie szybko wraz z obciążeniem systemu i stanowisko split-merge destabilizuje się przy wzroście strumienia nadchodzących zadań szybciej niż pozostałe typy stanowisk.

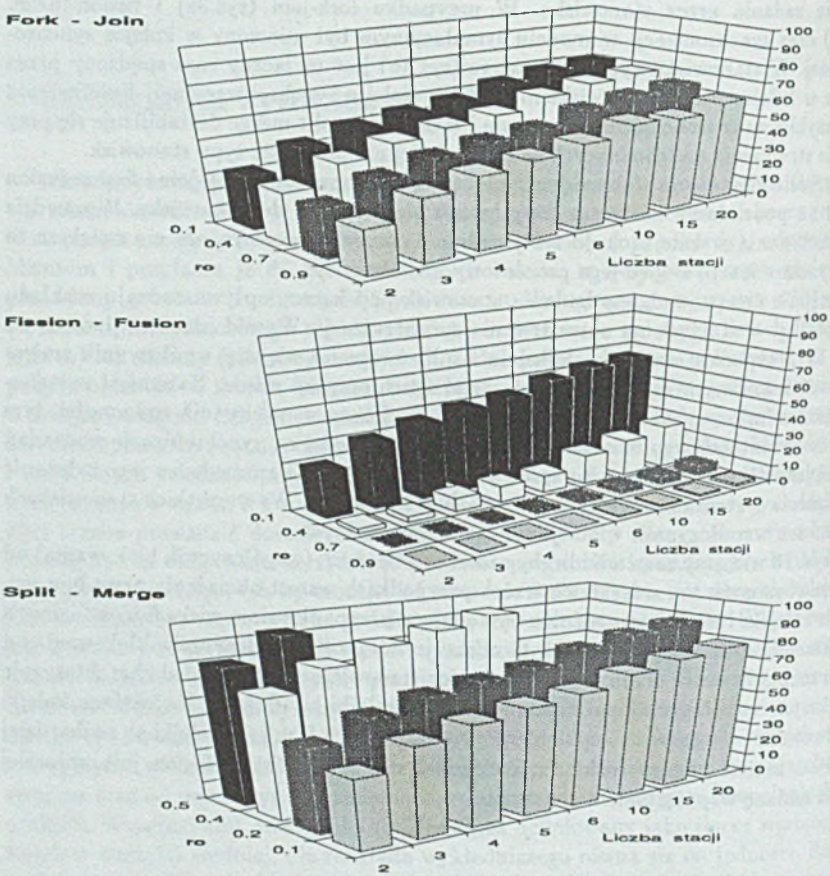
W przeciwieństwie do stanowiska split-merge, w stanowiskach fork-join i fission-fusion podział na podzadania następuje natychmiast po przyjściu do stanowiska. Wprawdzie czas oczekiwania w kolejkach do stacji rośnie wraz z obciążeniem, ale nie zwiększa to czasu synchronizacji, a więc jego procentowy udział maleje.

Następnie przeprowadzono badania stanowisk pod kątem wpływu rodzaju rozkładu czasu obsługi w stacjach na okres trwania synchronizacji. Wyniki przedstawione są na rys. 9. W przypadku stanowisk fork-join i split-merge im większy współczynnik zmienności w zastosowanym rozkładzie, tym udział synchronizacji rośnie. Natomiast w stanowisku fission-fusion sytuacja jest odwrotna – im większy współczynnik zmienności, tym udział czasu potrzebnego w całkowitym czasie odpowiedzi na synchronizację podzadań jest mniejszy. W badaniach nad współczynnikiem blokowania sprawdzono jego zależność od obciążenia systemu i rozkładu czasów obsługi w stacjach. We wszystkich stanowiskach mierzono ten współczynnik w kolejkach synchronizacyjnych.

Na rys. 10 widzimy zależność liczby zadań cząstkowych (współczynnik blokowania) od obciążenia systemu. We wszystkich trzech przypadkach wzrost obciążenia powoduje wydłużenie kolejek. Najmniejsze różnice występują w przypadku stanowiska fission-fusion, a największe dla split-merge. Rysunek 11 obrazuje zależność współczynnika blokowania od rodzaju rozkładu czasu obsługi. We wszystkich stanowiskach, im rozkład charakteryzuje się większym współczynnikiem zmienności, tym kolejki są dłuższe. Największe kolejki zaobserwowano w stanowisku split-merge. Na stanowisku fork-join kolejki są najbardziej zależne od zastosowanego rozkładu, natomiast stanowisko fission-fusion jest najmniej czułe na zmianę współczynnika zmienności.

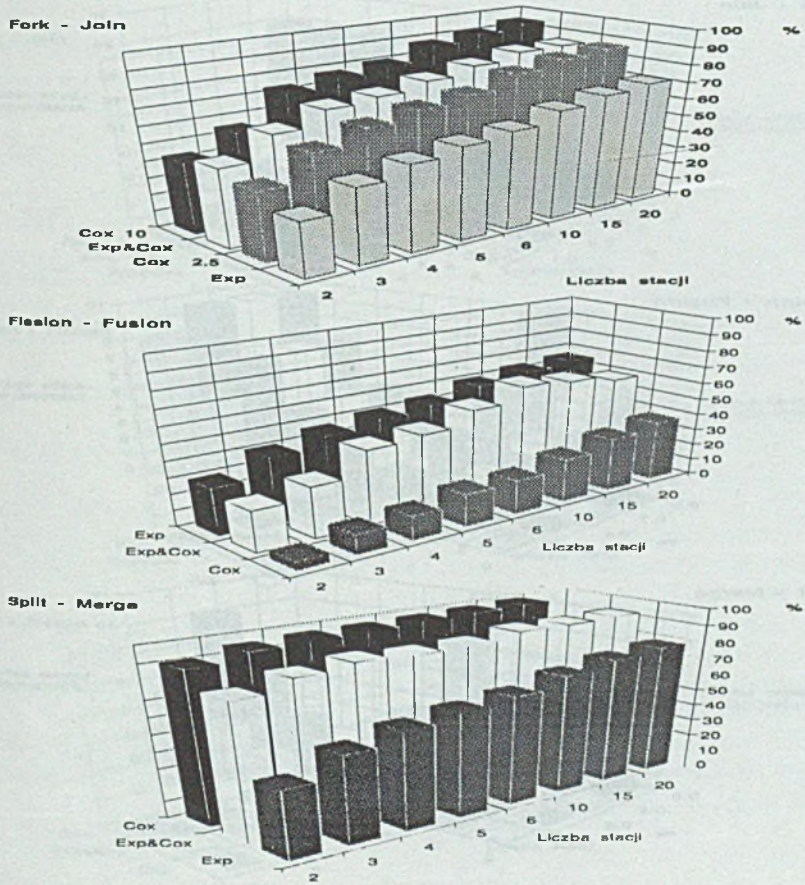
4. Wnioski

Problem poprawnego analitycznego modelu systemu fork-join i jemu pokrewnych, uwzględniającego dowolne rozkłady czasów obsługi i potrafiącego włączyć ten model do sieci stanowisk obsługi, w tym potrafiącego zagnieżdżać struktury fork-join wewnątrz nich samych, jest wciąż otwarty. Przedstawione modele analityczne i numeryczne stanowią rozwiązania wycinkowe, natomiast wyniki symulacji pozwalają zorientować się w charakterze wpływu poszczególnych parametrów modelu na sposób pracy stanowiska. Rezultaty wskazują, że przyspieszenia wynikające z równoległej organizacji pracy są w istotny sposób zmniejszone przez ograniczenia wnoszone przez synchronizację zadań.



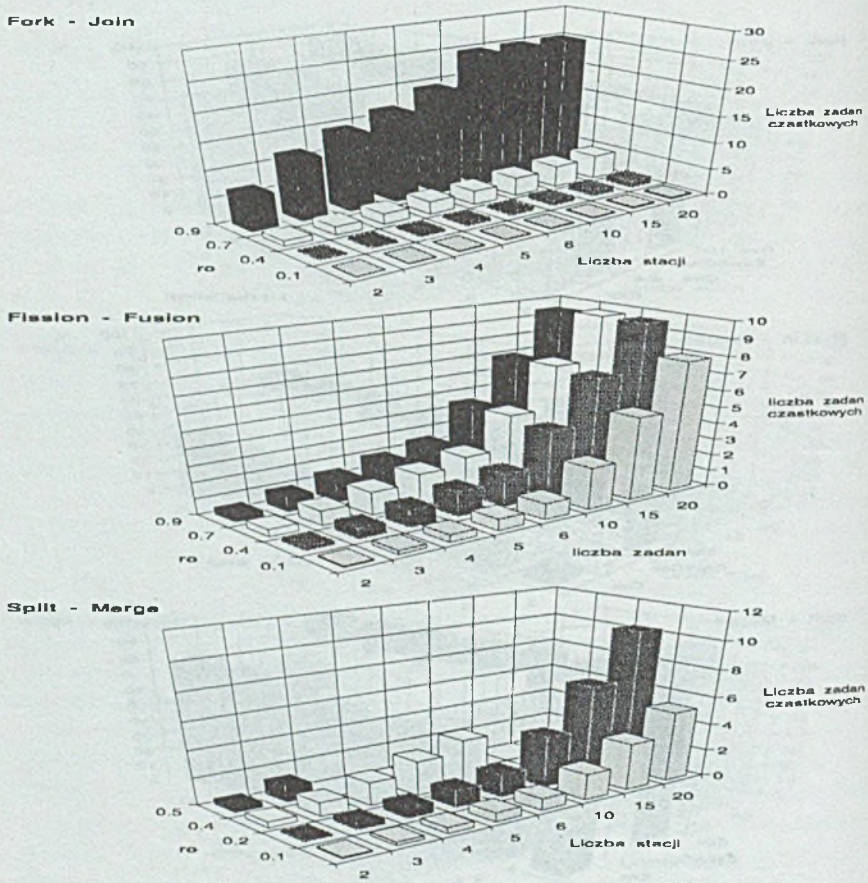
Rys. 8. Współczynnik synchronizacji w stanowiskach fork-join, fission-fusion i split-merge w funkcji obciążenia stanowisk; jednakowy wykładniczy rozkład czasu obsługi we wszystkich równoległych stanowiskach. Miejsca, w których brak wyników w stanowisku split-merge, odpowiadają przypadkom, w których to stanowisko destabilizuje się.

Fig. 8. Synchronization factor versus coefficient of utilisation in fork-join, fission-fusion and split-merge systems; the same exponential service time distribution in all parallel servers. Lacking results split-merge station correspond to the cases where station is a unstable.



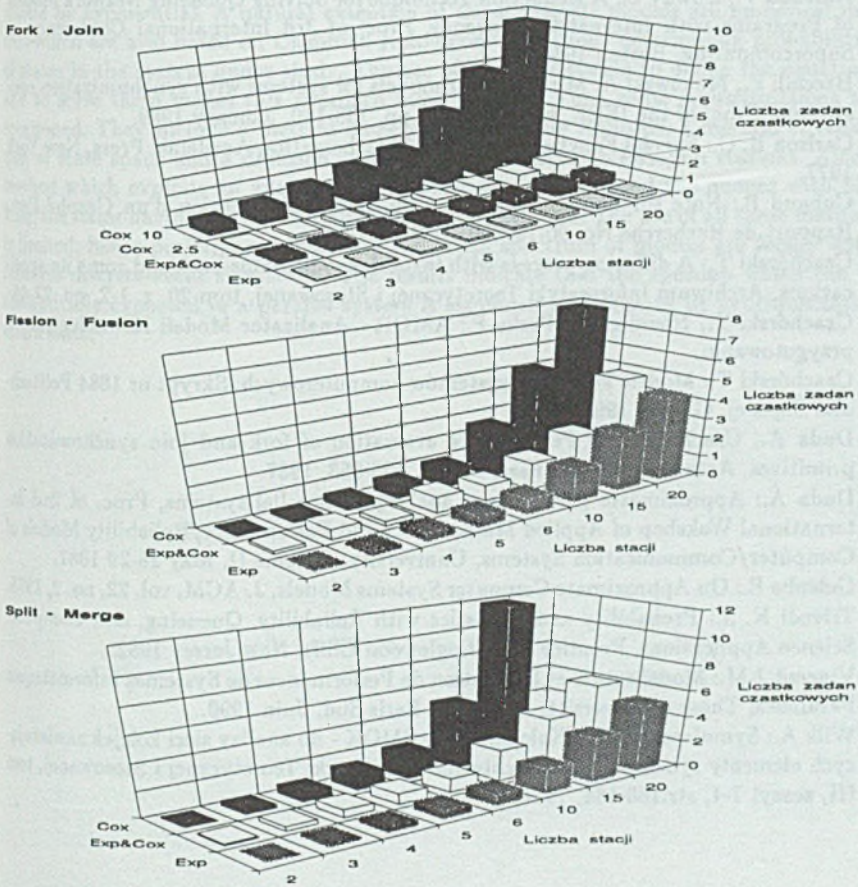
Rys. 9. Współczynnik synchronizacji w stanowiskach fork-join, fission-fusion i split-merge w funkcji współczynnika zmienności rozkładu czasu obsługi.

Fig. 9. Synchronization factor versus coefficient of variation of service time distribution in fork-join, fission-fusion and split-merge systems.



Rys. 10. Współczynnik blokowania w stanowiskach fork-join, fission-fusion i split-merge w funkcji obciążenia stanowiska; jednakowy wykładniczy rozkład czasu obsługi we wszystkich równoległych stanowiskach.

Fig. 10. Blocking factor versus coefficient of utilisation in fork-join, fission-fusion and split-merge systems; the same exponential service time distribution in all parallel servers



Rys. 11. Współczynnik blokowania w stanowiskach fork-join, fission-fusion i split-merge w funkcji współczynnika zmienności rozkładu czasu obsługi.

Fig. 11. Blocking factor versus coefficient of variation of service time distribution in fork-join, fission-fusion and split-merge systems

LITERATURA

[1] Almeida V., Dowdy L.: A Reduction Technique for Solving Queueing Network Models of Programs with Internal Concurrency, Proc. of 3rd International Conference on Supercomputing, Boston 1988.

[2] Baccelli F., Makowski A. M.: Queueing models for systems with synchronization constraints, Proc. of the IEEE, vol. 77, no. 1, pp. 138-160, January 1989.

[3] Carlson B. C.: Special Functions of Applied Mathematics, Academic Press, New York 1977.

[4] Cubaud P.: Note sur une approximation du Temps d'Execution d'un Graphe Pert, Rapport de Recherche No. 90-10, EHEI, Paris 1990.

[5] Czachórski T.: A diffusion process with instantaneous jumps back and some its applications, Archiwum Informatyki Teoretycznej i Stosowanej, tom 20, z. 1-2, pp. 27-46.

[6] Czachórski T., Niemiec M., Pecka P.: AMOR - Analizator Modeli Równoległych, w przygotowaniu.

[7] Czachórski T.: Modele kolejkowe systemów komputerowych, Skrypt nr 1884 Politechniki Śląskiej, Gliwice 1994.

[8] Duda A., Czachórski T.: Performance evaluation of fork and join synchronization primitives, Acta Informatica, Fasc. 24, pp. 525-553, 1987.

[9] Duda A.: Approximatte performance analysis of parallel systems, Proc. of 2nd International Wokshop of Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems, Unniversity of Rome II, May 25-29 1987.

[10] Gelenbe E.: On Approximate Computer Systems Models, J. ACM, vol. 22, no. 2, 1975.

[11] Trivedi K. S.: Probability and Statistics with Reliability, Queueing, and Computer Science Applications, Prentice Hall, Englewood Cliffs, New Jersey 1982.

[12] Vincent J-M.: Modélisation et Evaluation de Performances de Systèmes Informatiques Parallèles, Thèse d'Université, Université Paris-Sud, Juin 1990.

[13] Wilk A.: Symulator Modeli Kolejkowych - SMOK - do analizy sieci kolejek zawierających elementy synchronizacji, Archiwum Informatyki Teoretycznej i Stosowanej, tom III, zeszyt 1-4, str.165-184, 1991.

Recenzent: Prof. dr inż. Stefan Węgrzyn

Wpłynęło do Redakcji 27 grudnia 1995

Abstract

Fork-join, split-merge and *fission-fusion* models are a suitable tool to evaluate the efficiency of execution of jobs defined by parallel-sequential task precedence graph in multiprocessor environment. The article reviews existing methods which anlyse these

models. Analytic solutions aim to determine an equivalent server of the discussed model but the results are limited to 2 parallel servers in the case of fork-join and fission fusion models; in the case of split-merge model L servers are allowed. Service time distributions should be exponential. A natural extention of these analytic models are numerical models which are also based on Chapman-Kolmogorov equations determining probabilities of states in the system under study. The use of special software to define the equations and to solve them makes this approach more realistic. Then some approximations are mentioned. They include a method based on Mean Value Analysis, a method of reduction of state space and a diffusion approximation adopted to fork-join stations. Also a method which exploits an analogy between fork-join systems and the queues with bloking, the letter having known analytical solution is reviewed. The use of all these methods is limited, hence performance indices of the broad spectrum of models are finally obtained via discrete-event simulation. The results indicate that the speedup which can be theoretically expected in a parallel system is significantly decreased by synchronization constraints.