Dariusz DOROTA
Politechnika Krakowska

WYKORZYSTANIE SYSTEMÓW MASOWEJ OBSŁUGI DO SZEREGOWANIA ZADAŃ WIELOPROCESOROWYCH

Streszczenie. Zdefiniowano problem szeregowania wieloprocesorowych zadań (wymagających synchronicznego, równoczesnego użycia co najmniej dwóch procesorów do realizacji) z wykorzystaniem systemów masowej obsługi (SMO). Zaproponowano nowy rodzaj mieszanego systemu masowej obsługi wyposażonego w kanały komunikacyjne dedykowane dla zadań wieloprocesorowych. Docelowo zadania są szeregowane na architekturę NoC (*ang. Network on Chip*). Rozważono przypadki zadań: przerywalnych, nieprzerywalnych, zależnych oraz niezależnych. Zaadaptowano koncepcję dynamicznej zmiany zbioru zadań dla systemów masowej obsługi oraz zestawiono wyniki dla obu zaproponowanych podejść.

USING MASS SERVICE SYSTEMS TO SCHEDULE MULTIPROCESSOR TASKS

Summary. The problem of scheduling multiprocessor tasks (requiring synchronous simultaneous use of at least two processors for execution) using mass service systems is defined. A new type of mixed mass service system equipped with communication channels dedicated to multiprocessor tasks is proposed. Ultimately, tasks are scheduled on the NoC (Network on Chip) architecture. Cases of interruptible, uninterruptible, dependent and independent tasks are considered. The concept of dynamic change of the task set for mass service systems is adapted and results for both proposed approaches are compared.

1. Wstęp

Systemy wbudowane, IoT (*ang. Internet of Things*), IoV (*ang. Internet of Vehicle*), IoR (*ang. Internet of Robot*) są modelowane poprzez zagadnienia z obszaru szeregowania zadań. Zastosowanie wieloprocesorowych zadań jest motywowane podwyższeniem poziomu bezpieczeństwa, czy też niezawodności [2, 3] . Naturalnym wykorzystaniem wieloprocesorowości zadań jest redundancja sprzętowa, a także programowa. Wykonanie tego samego zadania (programu) na identycznych maszynach pozwala na podniesienie niezawodności oraz wiarygodności systemu. Systemy takiego typu stosowane są powszechnie w różnych dziedzinach takich jak: branża lotnicza, pojazdy kosmiczne, samochody, urządzenia o zastosowaniu militarnym, środki transportu materiałów niebezpiecznych, instalacje chemiczne, robotyka, drony czy też instalacje jądrowe [4, 5, 6]. Dla rozpatrywanego w ten sposób problemu została zaadaptowana koncepcja systemów

masowej obsługi [8]. Teoria kolejek, na której opiera się wymieniona koncepcja bazuje na statystyce matematycznej oraz rachunku prawdopodobieństwa do analizy zachowań napływających w sposób nieskończony zadań. Jednym z głównych celów systemu masowej obsługi jest minimalizacja czasu oczekiwania zgłoszenia na obsługę oraz innych wartości (np. koszt) [13, 12, 17, 1].

2. Sformułowanie problemu

Wprowadza się kilka oznaczeń, zbiór zadań jest sygnowany jako $T = \{T_1, T_2, T_3, ..., T_n\}$, te z kolei mogą być realizowane na identycznych równoległych maszynach zorganizowanych w architekturę sieci NoC. Procesory (maszyny) oznaczono jako $M = M_1, M_2, M_3, ..., M_m$. Czas trwania zadania $T_i \in T$ oznacza się jako p_i. A_i stanowi zbiór procesorów alokowanych równocześnie do realizacji zadania T_i. Pojedyncze zadanie wymaga do wykonywania synchronicznego (jednoczesnego) dostępu do $a_i \geqslant 1$ procesorów. Graf $G = (T, E)$ jest acykliczny, gdzie $E \subseteq 2^V$ jest zbiorem nieskierowanych krawędzi. Krawędź jest reprezentowana zbiorem dwuelementowym $\{i, j\}$, $i, j \in V$. W niektórych przypadkach G może mieć specyficzną postać, np. drzewa. Harmonogram realizacji zadań jest opisany poprzez parę wektorów (S, A), gdzie $S = (S_1, ..., S_n)$, S_i jest terminem rozpoczęcia zadania T_i; $A = (A_1, ..., A_n)$, $A_i \subseteq M$, $|A_i| = a_i$. Należy wyznaczyć harmonogram pracy procesorów, tak by spełnione były powyższe ograniczenia oraz minimalizowany był wskaźnik jakości, przyjęty jako długość uszeregowania (*ang. makespan*) $C_{\max} = \max_{1 \leqslant i \leqslant n}(S_i + p_i)$. Rozpatruje się zadania przerywalne oraz nieprzerywalne. W pierwszym przypadku zakłada się, że zadanie jest podziolne na sekwencję pewnej skończonej liczby operacji x_i. Oznaczając to jako operacje $O_i = (o_{i,1}, o_{i,2}, o_{i,3}, ..., o_{i,x_i})$, wykonywanych w ustalonej kolejności, przy założeniu, że każda wymaga użycia a_i procesorów. Musi zachodzić $\sum_{k=1}^{x_i} |o_{i,k}| = p_i$, gdzie $|o_{i,k}|$ oznacza czas trwania operacji $o_{i,k}$. W takim przypadku podział zadania na operacje podlega wyborowi.

Praca niniejsza jest rozszerzeniem poprzednich wątków w tym zakresie poruszanych przez autora. Systematyzując problemy szeregowania należy wspomnieć o taksonomii $\alpha|\beta|\gamma$. Istotnym okazała się konieczność jej rozszerzenia (symbol β) o oznaczenia zadań wieloprocesorowych jako a_i. Wartość ta może być stałą $(a_i = k)$ lub zmienną $a_i \leqslant k$, co oznacza odpowiednio: A) wszystkie zadania są k-procesorowe, B) wszystkie zadania mogą mieć współczynnik wieloprocesorowości o przedziale od $1..k$. Wykorzystując oznaczenia ww. notacji trójpolowej można rozważane problemy przedstawić jako $P|\beta|C_{max}$, gdzie $\beta \in \{\circ, a_i = k, a_i <= k, a_i, pmtn, prec, smo(p)\}$ [18, 15, 19, 10, 11, 16, 8]. W tym przypadku do procesu szeregowania zaproponowano wykorzystanie systemów masowej obsługi, opierających się na teorii kolejek. Zaproponowano mieszany system obsługi z oddzielnymi kanałami obsługi dla każdego z rodzajów zadań (w zależności od współczynnika a_i). W systemach kolejkowych dane są zmiennymi losowymi, zatem analizę prowadzi się dla miar probabilistycznych (np. wartości średnie) jak intensywność napływu zgłoszeń, prawdopodobieństwo zajętości stanowiska, średni czas oczekiwania na wykonanie zadania, itp.

3. Systemy masowej obsługi

Głównym celem w obszarze systemów wbudowanych jest minimalizacja, co najmniej jednego z parametrów takich jak: czas oczekiwania na obsługę, koszty skorelowane z wyżej wymienionymi zadaniami. Systemy masowej obsługi scharakteryzować można z wykorzystaniem elementarnych pojęć dedykowanych dla tego typu zagadnienia: A) Zgłoszenie - obiekt oczekujący na obsługę, B)Kanały obsługi - element obsługujący zgłoszenia, C) Kolejka - zbiór zgłoszeń oczekujących na obsługę, D) Obsługa - czynność pozwalająca na zaspokojenie określonych potrzeb, E) Regulamin - reguły dotyczącej zgłoszeń obsługiwanych i oczekujących na obsługę.

Systemy kolejkowe opisać można z wykorzystaniem notacji Kendall'a w postaci X/Y /m gdzie: X - rozkład odstępów czasowych pomiędzy zgłoszeniami, Y - rozkład czasów obsługi, m - liczba kanałów. Ta podstawowa klasyfikacja okazała się zawierać pewne niedogodności i w celu ich eliminacji zaproponowane jest rozszerzenie w notacji Lee, która używa zapisu X/Y /m/d/l. Poszczególne symbole dla obu wymienionych notacji oznaczają odpowiednio: X- rozkład czasowy pomiędzy kolejnymi zgłoszeniami, Y - rozkład czasów obsługi, m - liczba kanałów, d - kod polityki kolejki, l - wymiarowość systemu. Symbole X, Y mogą przyjmować następujące wartości określające rozkłady prawdopodobieństwa: D - deterministyczny, M- wykładniczy, E_k - Erlanga rzędu k, Hr - hiperwykładniczy rzędu r, C_k - Cox'a rzędu k, GI - dowolny i niezależny, G - dowolny. Oprócz tej podstawowej klasyfikacji można także dokonać rozróżnienia pod względem sposobu tworzenia kolejki: A) zabroniona, B) dozwolona, C) ze stratami, D) bez strat. Kolejny sposób klasyfikacji uwzględnia priorytetyzację kolejki (sposób bardziej praktyczny podział niż ten zaproponowany przez Erlanga) zakłada występowanie kolejek o regułach: FIFO (*ang. First in First Out*), SIRO (*ang. Selection in Random Order*), LIFO (*ang. Last in First Out*).

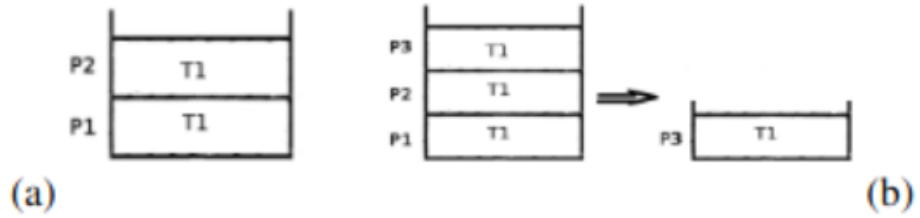
W systemach kolejkowych dane zadania są zmiennymi losowymi, zatem analizę prowadzi się dla miar probabilistycznych, przykładowo wartości średniej wielkości, szerzej przedstawione w pracach [14, 1, 17, 12, 13]:

- intensywności napływu zgłoszeń (λ) oraz obsługi zgłoszeń (μ), gdzie τ_λ - średni czas pomiędzy zgłoszeniami, τ_μ - średni czas obsługi zgłoszenia,
- p - prawdopodobieństwo zajętości stanowiska,
- w_i - średni czas oczekiwania na wykonanie zadania o priorytecie i dla zadań z wywłaszczaniem, gdzie m_{2j} (patrz [68]),
- S_i - czas potrzebny do obsłużenia zadań o wyższych priorytetach niż zadania realizowanego
- q_i - średni czas przebywania zadania i w systemie

W niniejszym artykule zaproponowano system kolejkowy zamknięty, z różnymi strumieniami zgłoszeń, z obsługą wielokanałową synchroniczną, lub nietypową politykę obsługi w oparciu o zasoby. Koncepcja taka została zaproponowana w [8]. Dodatkowo rozszerzono zaproponowany model o obsługę dynamicznego zbioru zadań, również zaproponowaną w pracy [9].

4. Zadania wieloprocesorowe

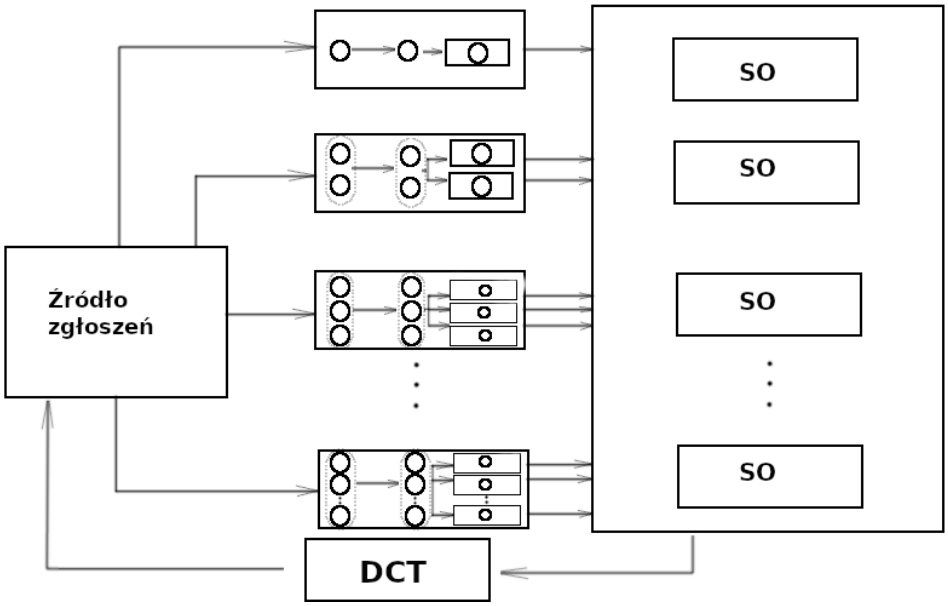
W celu zwiększenia niezawodności systemu wprowadzana jest redundancja sprzętowa i programowa poprzez synchroniczne wykonywanie identycznych zadań, na co najmniej dwóch identycznych procesorach pracujących równolegle. Zaproponowana redundancja znajduje zastosowanie takie jak: jednoczesny dostęp do pamięci współdzielonej, testowanie układów VLSI (*ang. Very Large Scale Integration*), czy też realizacji prawdziwej równoległości. Zasadniczo koncepcję zobrazowano na rysunku 2, gdzie ukazano propozycję zastosowania zadań wieloprocesorowych. Skupiono się na przykładach najbardziej oczywistych dla zadań dwu- oraz trzy-procesorowych, co pokazano na rysunku 1. Zakłada się, że jeżeli odpowiedzi dla zadania wieloprocesorowego są identyczne (dla każdego z procesorów), to zadanie jest uważane za prawidłowo wykonane. Zarówno wykonanie zadań dwu jak i trzy procesorowych pozwala na zwiększenie niezawodności systemu [10]. W praktyce tylko zadania o krytycznym znaczeniu dla systemu są realizowane jako wieloprocesorowe. Stąd jest zasadne rozpatrywanie problemów $a_i = k$ oraz $a_i \leqslant k$.



Rys. 1. Koncepcja i motywacja wykorzystania zadań wieloprocesorowych, źródło [8]

5. Koncepcja systemu

Punktem wyjścia do dalszych rozważań jest koncepcja algorytmu deterministycznego. Ten bazujący na koncepcji Muntza-Coffmana algorytm szereguje zadania wieloprocesorowe zarówno zależne jak i niezależne. Dalsze prace autora skupiały się na wprowadzeniu niepewności do procesu szeregowania zadań wieloprocesorowych, adaptując odpowiednio logikę rozmytą, stochastykę, czy też systemy online. Jednym z rozszerzeń wykorzystujących parametr niepewności jest koncepcja bazująca na systemach masowej obsługi, która pozwalaja na szeregowanie zadań wieloprocesorowych. Zaproponowano model systemu mieszanego rozszerzony o obsługę zadań wieloprocesorowych. Można zauważyć, że każdy kanał jest odpowiedzialny za obługę zadań o określonym współczynniku a_i, gdzie zadania są dodawane do kolejek w odpowiednich kanałach. Następnie w zależności od wyznaczonego parametru ready time są szeregowane na docelowej architekturze. Przy czym dla zadań dwu-procesorowych istnieje możliwość ponownego wykonania zadania (jako zadania trzy-proceosorowego) jeżeli odpowiedzi dla zadania dwu-proceosorwego są różne dla obu procesorów. Stanowi to połączenie koncepcji dynamicznej zmiany zbioru zadań (przedstawionej w artykule [11]) z systemami masowej obsługi. Poniżej zaprezentowano koncepcję omawianego systemu w formie graficznej.



Rys. 2. Koncepcja wykorzystania SMO do szeregowania zadań wieloprocesorowych, źródło opracowanie własne na podstawie [8]

6. Algorytmy

Zaproponowane podejście wykorzystujące systemy kolejkowe rozstrzyga obsługę zadań wieloprocesorowych, wykorzystując pewną modyfikację algorytmu MC, z wcześniejszych prac autora [9, 8, 7, 6, 5].

Zastosowane oznaczenia dla problemu szeregowania online: $ReadyTime$ - termin gotowości zadania T_i, $w_i = p_i \cdot a_i$ priorytet zadania T_i, d_i - najpóźniejszy możliwy czas rozpoczęcia zadania T_i, $D_i = d_i + p_i$ - najpóźniejszy możliwy termin zakończenia zadania T_i (dwa ostatnie oznaczenia są opcjonalnie określane w specyfikacji). Algorytm 1 dla szeregowania zadań wieloprocesorowych wykorzystujących systemy masowej obsługi oraz, koncepcję dynamicznego zbioru zadań, przedstawiono w formie pseudokodu. Kolejność wyznaczania i analizowania priorytetów zależy od zadań i grafu G oraz jego struktury. Zależność zadań reprezentowana grafem G jest oznaczona w pseudokodzie jako TG. Opis znaczenia kroków algorytmu podano w komentarzach.

Algorytm 1. Algorytm MC - SMO:

1. $t = 0$;
2. for $T_i \in T$ do oblicz w_i;
3. for $T_i \in T$ do oblicz t_i;
4. for $T_i \in T$ do oblicz h_i;
5. if $E \neq \emptyset$ then
6. for $T_i \in T$ do $w_i = w_i + t_i$; /*korekta dla zadań zależnych*/
7. end if;

8. for $T_i \in T$ do if $t_i > D_i$ then stop: uszeregowanie nie istnieje;
9. P=T; /* kopia T */
10. for $T_i \in T$ do $q_i = p_i$; /* kopia p_i */
11. while $P \neq \emptyset$ do
12. av=m;
13. $Q = \{T_i \in P : \ (q_j = 0, \ j \in B_i) \wedge (q_i > 0)\}$; /*zbiór zadań gotowych*/
14. if $Q = \emptyset$ then goto end-while;
15. $M = \max_{T_j \in Q} w_j$; /*maksymalny priorytet w Q*/
16. $W = \{T_j \in Q : \ w_j = M\}$; /*znajdź zadania z priorytetem M*/
17. $A = \max_{T_j \in W} a_j$; /*maksymalne a_j w W*/
18. $F = \{T_j \in W; \ a_j = A\}$; /*zadania z W z żądaniem A*/
19. $C = \max_{T_j \in F} h_j$; /*maksymalne h_j w F*/
20. $J = \{T_j \in F; \ h_j = C\}$; /*zadania z F o $h_j = C$*/
21. wybierz dowolne $T_i \in J$;
22. $K_j = K_j + T_i$ /* Do odpowiedniej kolejki (obsługującej zadania, gdzie $a_k = j$ dodaj zadanie T_i)*/
23. for $K_j \in K$,
24. for $T_j \in K_j$
25. $x = 1$;
26. if $A \leqslant av$ then
27. uszereguj T_i dla jednej jednostki czasowej $[t, t+1]$;
28. $av = av - a_i$; $q_i = q_i - x$; $Q = Q - \{T_i\}$;
29. if $(a_i == 2)$ $find(x_j)$
30. if $(a_i == 2)$ and $(x_j == 1)$
31. dodaj do zbioru T dodatkowe zadanie ($T\prime_j$ gdzie $a_j = 3$)
32. end-if
33. else
34. $Q = Q - \{T_i\}$;
35. end if-else;
36. if $av > 0$ go to step 14; /* gdy są jeszcze wolne procesory w chwili t*/
37. else
38. $t = t + x$;
39. end if-else;
40. if $q_i = 0$ then $P = P - T_i$;

41. end for;
42. end for;
43. end while;

Opis algorytmu:

- Specyfikację dla systemu stanowi graf zadań, w którym wprowadzono oznaczenia dla zadań wymagających do wykonania więcej niż jednego procesora (zadania wieloprocesorowe są wybierane w sposób losowy). Mogą występować relacje poprzedzeń pomiędzy zadaniami. Zadania są przekazywane do odpowiedniego kanału obsługi w zależności od współczynnika a_i, przy czym dla każdej z wartości współczynnika wieloprocesorowości istnieje odrębny kanał obsługi.
- Ze zbioru priorytetów wybierane są zadania o najwyższych wartościach, uwzględniając położenie każdego zadania na ścieżce (o ile taka występuje).
- Stosując się do powyższych zasad, wybierane są zadania z określonymi priorytetami i następnie zostają alokowane w wybranych procesorach, tak aby nie występowały niewykorzystane w jednostce czasu procesory.
- Jeżeli w jednostce czasu nie występują bezczynne procesory, lub niemożliwe jest przypisanie zadania do niewykorzystanych procesorów, to czas przesuwany jest o kolejną jednostkę czasu, dopóki nie zostaną uszeregowane wszystkie zadania.
- Jeżeli przypisane do procesora zadanie dwu-procesorowe wygeneruje inny wynik na tych maszynach, to zadanie ponownie jest przekazywane do źródła zgłoszeń, tym razem jako zadanie trzy-procesorowe.

7. Otrzymane rezultaty

Przeprowadzone zostały badania eksperymentalne dla opisanego podejścia. Zestawiono je w tabeli, dodatkowo porównując z wcześniejszymi rezultatami otrzymanymi przez autora w pracy [8]. Eksperymenty przeprowadzono dla wygenerowanych zadań i przedstawiono w tabeli 1, gdzie poszczególne kolumny oznaczają odpowiednio nr zadania (Nr.), czas trwania zadania (p_i), współczynnik wieloprocesorowości (a_i), priorytet zadania (w_i), czas gotowości (Ready Time), najpóźniejszy możliwy czas zakończenia zadania (Finish time), średni czas wykonania zadania, konieczność ponownego wykonania zadania (w przypadku zadań dwu-procesorowych, o ile prawdopodobieństwo otrzymania różnych odpowiedzi od procesorów wykonujących zadania jest bliskie 1 (w praktyce przyjęto wartość większą lub równą 0,7). Fakt ten oznacza konieczność ponownego wykonania zadania, przy czym angażowana jest większa liczba procesorów do jego realizacji, co nazwano dynamiczną zmianą zbioru zadań, DCT (*ang. Dynamic Change Task*).

Rezultaty przedstawione w tabeli 1 zilustrowano na rysunku 3

8. Podsumowanie

Zaproponowane zostało podejście do szeregowania zadań wieloprocesorowych wykorzystujące koncepcję systemów masowej obsługi w połączeniu z dynamiczną

Tabela 1. SMO z wykorzystaniem koncepcji DCT, $m = 5$, $n = 50$

Nr.	p_i	a_i	w_i	Ready Time	Finish Time	Avg. Time	AVG	Avg. Time (DCT)
T0	23	4	92	0	24	24	24,00	24,00
T19	14	2(3)	28	52	66	14	19,00	19,00
T45	14	1	14	65	79	14	17,33	17,33
T28	5	2(3)	10	71	81	10	15,50	16,50
T1	13	3	39	120	132	12	14,80	15,20
T46	6	4	24	124	157	33	17,83	14,33
T49	18	4	72	138	156	18	17,86	14,00
T15	28	1	28	147	175	28	19,13	16,38
T44	6	2(3)	12	150	163	13	18,44	16,56
T4	26	4	104	175	201	26	19,20	17,70
T0	23	4	92	0	24	24	24,00	17,27
T19	14	2(3)	28	52	66	14	19,00	16,92
T45	14	1	14	65	79	14	17,33	17,62
T28	5	2	10	71	81	10	15,50	16,43
T1	13	3(3)	39	120	132	12	14,80	16,87
T46	6	4	24	124	157	33	17,83	16,00
T49	18	4	72	138	156	18	17,86	15,47
T15	28	1	28	147	175	28	19,13	16,33
T44	6	2(3)	12	150	163	13	18,44	15,89
T4	26	4	104	175	201	26	19,20	16,05
T0	23	4	92	0	24	24	24,00	16,57
T19	14	2(3)	28	52	66	14	19,00	17,00
T45	14	1	14	65	79	14	17,33	21,70
T28	5	2(3)	10	71	81	10	15,50	22,08
T1	13	3	39	120	132	12	14,80	21,68

Rys. 3. Średni czas przepływy dla podejścia SMO z DCT

zmianą zbioru zadań. Przeprowadzone eksperymenty ukazują zasadność zaproponowanego podejścia. Można zaobserwować (rys. 3) tendencję do stabilizacji systemu (średniego przepływu). Jest to rezultat podobny do tego otrzymanego w pracy [8], przy czym w przypadku niniejszej pracy dodana koncepcja "ponownego sprawdzenia poprawności wykonania zadania" wpływa na podwyższenie niezawodności całego systemu, nieznacznie przy tym zmieniając tendencję do ustabilizowania się systemu. Otrzymane rezultaty

pozwalają stwierdzić, że dodatkowo zaproponowany w algorytmie aspekt podniesienia niezawodności (poprzez ponowne wykonanie wybranych zadań, zgodnie z regułami), nie tylko nie wpływa negatywnie na średni czas przepływu, ale także poprawia samą niezawodność systemu. Naturalnym kierunkiem wydaje się uwzględnienie czasów transmisji pomiędzy poszczególnymi zadaniami.

LITERATURA

1. Adan, I., & Resing, J. (2015). Department of Mathematics and Computing Science Eindhoven University of Technology PO Box 513, 5600 MB Eindhoven, The Netherlands March 26, 2015.
2. Błażewicz, J. et al.: Handbook on scheduling: from theory to applications, Springer Science & Business Media, 2007.
3. Błażewicz J. and Liu Z.: Scheduling multiprocessor tasks with chain constraints, European Journal of Operational Research, Elsevier, 1996.
4. Chen X.: Selected problems of online scheduling on parallel machines, Rozprawa doktorska, Politechnika Poznańska, Poznań, 2014.
5. Dorota D.: Dual-processor tasks scheduling using modified Muntz-Coffman algorithm. In: International Conference on Dependability and Complex Systems. Springer, Cham, 2018.
6. Dorota D.: Scheduling Tasks in a System with a Higher Level of Dependability. In: International Conference on Dependability and Complex Systems. Springer, Cham, 2019.
7. Dorota D.: Scheduling tasks with uncertain times of duration. In: International Conference on Dependability and Complex Systems. Springer, Cham, 2020.
8. Dorota D. : Szeregowanie zadań wieloprocesorowych w warunkach niepewności, Politechnika Wrocławska, 2023.
9. Dorota, D. (2023). Dynamic Change of Tasks in Multiprocessor Scheduling. In International Conference on Dependability and Complex Systems (pp. 39-50). Cham: Springer Nature Switzerland.
10. Drozdowski M.: Scheduling for parallel processing. Springer, London, 2009.
11. Drozdowski M.: Scheduling multiprocessor tasks - an overview, European Journal of Operational Research, Elsevier, 1996.
12. Ficoń, K.: Wykorzystanie teorii kolejek w procesie decyzyjnym przedsiębiorstwa transportowego. Systemy Logistyczne Wojsk, 2019, vol.50, p. 65-78.
13. Gaj, P. (2016). Wybrane zagadnienia projektowania systemów informatyki przemysłowej. Studia Informatica, 37(4B), 1-400.
14. Harchol-Balter, M. (2013). Performance modeling and design of computer systems: queueing theory in action. Cambridge University Press.
15. Johannes B.: Scheduling parallel jobs to minimize the makespan, Journal of Scheduling, Springer, 2006.

16. Makuchowski M.: Problemy gniazdowe z operacjami wielomaszynowymi : Własności i algorytmy. Rozprawa doktorska. Raporty Instytutu Cybernetyki Technicznej PWr. 2004, Ser. PRE; nr 37. p. 196
17. Oniszczuk, W. (1995). Metody modelowania. Wydaw. Politech. Białostockiej.
18. Pinedo M. : Scheduling. Springer, New York, NY, 2015.
19. Smutnicki C. : Algorytmy szeregowania zadań. Oficyna Wydawnicza Politechniki Wrocławskiej, 2012.