Przetwarzanie równoległe i rozproszone - zadania egzaminacyjne

1. Porównaj przetwarzanie: współbieżne, równoległe i rozproszone (definicje, sprzęt, cechy charakterystyczne, dziedziny zastosowań)

2. Czym jest wyścig (*race condition*) w przetwarzaniu współbieżnym. Podaj przykład wyścigu prowadzącego do niedeterministycznego działania programu. W jaki sposób można wprowadzać synchronizację działania wątków i procesów. Zdefiniuj i omów następujące pojęcia: wzajemne wykluczanie, sekcja krytyczna, operacja atomowa. Jakie problemy pojawiają się, kiedy wprowadza się mechanizmy synchronizacji? Co to jest bezpieczeństwo i żywotność w odniesieniu do przetwarzania współbieżnego?

3. Opisz sposób działania zamka (*lock*) w postaci zwykłej zmiennej globalnej. Podaj wady stosowania takich zamków. Przedstaw definicję i sposób działania semafora (abstrakcyjnego – nie np. Unixowego). W jaki sposób semafor poprawnie rozwiązuje problem wzajemnego wykluczania. Czym jest uczciwość semafora i co o niej decyduje?

4. Scharakteryzuj problem ucztujących filozofów. Podaj jego rozwiązania za pomocą semaforów: jedno dopuszczające blokady i drugie poprawne (algorytmy powinny zawierać komentarze wyjaśniające co dzieje się w kolejnych linijkach). Wytłumacz kiedy dochodzi do blokady w pierwszym algorytmie i jak unika się blokady w drugim.

5. Czym są procesy, a czym wątki – jakie są między nimi różnice w definicjach i sposobie funkcjonowania. Podaj typową postać (w języku C lub pseudokodzie) dwóch programów w środowisku UNIXowym, jednego, w którym tworzony jest nowy proces realizujący odrębny program (zapisany w pliku wykonywalnym na dysku) oraz drugiego, w którym tworzony jest nowy wątek realizujący pewną procedurę. Wyjaśnij znaczenie argumentów i zwracanego wyniku w wywołaniu procedury:

```
int   pthread_create( pthread_t *thread,   pthread_attr_t *attr,   void(*start_routine)(void *),   void * arg )
```

6. Czym jest muteks wg specyfikacji POSIX, podaj podstawowe operacje związane z muteksami. Przedstaw przykład programu, w którym zmienna muteks wykorzystana jest do rozwiązania problemu wzajemnego wykluczania (algorytm powinien zawierać komentarze wyjaśniające co dzieje się w kolejnych jego linijkach). Uzasadnij dlaczego w programie konieczne jest zastosowanie wzajemnego wykluczania.

7. Czym są i do czego służą zmienne warunku (*condition* variables). W jaki sposób zmienne warunku uogólniają działanie semaforów. Przedstaw przykład programu, w którym zastosowane są zmienne warunku (np. rozwiązanie problemu producentów i konsumentów lub implementację bariery – algorytm powinien zawierać komentarze wyjaśniające co dzieje się w kolejnych jego linijkach). Podaj operacje realizowane przez system zarządzania wątkami w wyniku wywołania następujących procedur związanych ze

zmiennymi warunku: pthread_cond_wait(&zmienna_warunku, &muteks); pthread_cond_signal(&zmienna_warunku);

8. Omów model obliczeń równoległych z wykorzystaniem pamięci wspólnej. Przedstaw (najlepiej w postaci ilustracji z omówieniem) schemat obliczeń równoległych w modelu OpenMP z uwzględnieniem następujących elementów: wątek główny, obszar równoległy, komunikacja, bariera, obszar sekwencyjny. Co dzieje się w trakcie wykonania programu po napotkaniu dyrektywy *parallel?* W jaki sposób można określić liczbę wątków w obszarze równoległym?

9. Omów części składowe środowiska programowania OpenMP. Wymień podstawowe grupy dyrektyw i opisz jaka jest rola poszczególnych grup i pojedynczych dyrektyw (w jednym zdaniu na grupę lub dyrektywę). Wymień podstawowe grupy klauzul i opisz jaka jest rola poszczególnych grup lub klauzul (w jednym zdaniu na grupę lub klauzulę). Jakie znasz zmienne środowiskowe wprowadzane przez OpenMP i jaka jest ich rola oraz sposób funkcjonowania. Wymień i krótko scharakteryzuj podstawowe funkcje (grupy funkcji) z biblioteki OpenMP.

10. Wymień i krótko scharakteryzuj dyrektywy podziału pracy w OpenMP. Dla każdej dyrektywy przedstaw fragment kodu zawierający ją i omów w jaki sposób program będzie wykonywany wielowątkowo. Omów szczegółowo dyrektywę #pragma omp for uwzględniając klauzulę schedule określającą sposób podziału iteracji między wątki. Przedstaw sposób realizacji równoległej pętli w języku C (sposób realizacji oznacza przydział poszczególnych iteracji kolejnym procesorom maszyny):

```
#pragma omp parallel for klauzula
        for(i=0;i<20;i++) A[i]=0.0;
```

dla maszyny czteroprocesorowej i następujących przypadków *klauzuli*:

a) schedule (static)
b) schedule (static,4)
c) schedule (static,20/4)
d) schedule (dynamic)
e) schedule (dynamic,4)
f) schedule (dynamic,20/4)

11. Przedstaw i omów klauzule współdzielenia zmiennych w OpenMP. Dla każdej grupy zmiennych podaj, w którym obszarze pamięci będą przechowywane i jak ewentualnie będą traktowane ich wartości na początku i końcu realizacji dyrektywy. Zwróć szczególną uwagę na klauzulę *reduction* – podaj i szczegółowo objaśnij przykład obliczeń z wykorzystaniem tej klauzuli. Podaj jak traktowane są domyślnie różne grupy zmiennych w ramach obszarów równoległych (zmienne deklarowane na zewnątrz obszaru, wewnątrz, zmienne sterujące pętli itp.).

12. Omów jakie są podstawowe typy zależności między instrukcjami (w kontekście obliczeń równoległych) i na czym one polegają? Opis zależności zilustruj przykładami kodu (fragmenty kody bez pętli, przykłady muszą być inne niż na wykładzie). Pokaż w jaki sposób można usunąć niektóre z zależności. Podaj przykłady różnych typów zależności danych **przenoszonych przez pętle** (przykłady także inne niż na wykładzie). Przedstaw sposoby (o ile istnieją) modyfikacji kodu umożliwiające pozbycie się zależności i wykonanie równoległe.

13. Scharakteryzuj ogólnie model programowania równoległego z przesyłaniem komunikatów. Jak funkcjonują programy MPMD i SPMD? Podaj standardowe formy dwóch podstawowych operacji komunikacyjnych: send i receive. Omów ich argumenty - co oznaczają i jaka jest ich rola. Omów różnicę między przesyłaniem i odbieraniem blokującym i nieblokującym. Jakie dodatkowe procedury konieczne są do realizacji procedur nieblokujących. Wyjaśnij znaczenie argumentów i funkcjonowanie procedury:

```
int MPI_Wait(MPI_Request *preq, MPI_Status *pstat)
```

14. Wymień i scharakteryzuj sześć podstawowych procedur MPI tworzących podstawę do tworzenia programów równoległych. Podaj przykład prostego programu wykorzystującego te procedury (program w stylu SPMD, w którym używa się rangi procesu i rozmiaru komunikatora do podziału pracy między procesory; składnia wywołań nie musi być bezbłędna). Wyjaśnij znaczenie argumentów i funkcjonowanie procedur:

```
int  MPI_Send( void* buf,  int count,  MPI_Datatype dtype,  int dest,  int tag,  MPI_Comm comm )
int  MPI_Recv(void *buf, int count, MPI_Datatype dtype, int src, int tag, MPI_Comm comm, MPI_Status *stat)
```

15. Czym są obiekty nieprzenikalne w MPI i w jaki sposób funkcjonują, podaj przykłady takich obiektów. Wyjaśnij pojęcia: komunikator, interkomunikator, ranga, grupa odległa procesów, pamięć odległa. Scharakteryzuj różne tryby wysyłania komunikatów dwupunktowych (buforowany, synchroniczny, itd.).

16. W jaki sposób realizowany jest w MPI-2 bezpośredni dostęp procesorów do pamięci odległej? Podaj przykład programu, w którym realizowana jest komunikacja jednostronna z dostępem do pamięci odległej (algorytm powinien zawierać komentarze wyjaśniające co dzieje się w kolejnych jego linijkach).

17. Przedstaw interfejs równoległych operacji wejścia/wyjścia w MPI-2. Czym są typ plikowy i widok pliku. W jaki sposób można dokonywać równoległego odczytu i zapisu do plików w ramach specyfikacji MPI-2 (wykorzystanie jawnego i niejawnego offsetu). Podaj dwa przykłady programów MPI z równoległymi operacjami wejścia/wyjścia – jeden z jawnym offsetem i drugi z niejawnym (algorytmy powinny zawierać komentarze wyjaśniające co dzieje się w kolejnych ich linijkach).

18. W jaki sposób MPI umożliwia dynamiczne zarządzanie procesami? Jak następuje nawiązanie komunikacji pomiędzy procesami „rodzicami" i procesami „dziećmi". Jaka jest rola procedury MPI_Comm_get_parent. Wyjaśnij znaczenie argumentów i funkcjonowanie procedury:

```
int MPI_Comm_spawn( char* command, char* argv[], int maxprocs, MPI_Info info, int root,  MPI_Comm
    comm, MPI_Comm* intercomm, int array_of_errcodes[]);
```

Przedstaw przykład programu „rodzica" i programu „dziecka" (**dwóch różnych** programów) z wzajemną wymianą komunikatów.

19. Z czego składa się typ danych MPI (jaka jest jego abstrakcyjna definicja). Narysuj mapę pamięci oraz przedstaw definicję (jako zbiór par: przesunięcie,typ) dla typu danych MPI określonego za pomocą następujących struktur danych i procedur (należy narysować fragment pamięci - zaznaczając np. jeden bajt jedną komórką i wypełniają zgodnie z poniższym kodem):

a)

```
int tab_dlug_blokow[3] = {1, 1, 3 };
MPI_Datatype tab_typow[3] = { MPI_DOUBLE, MPI_CHAR, MPI_FLOAT };
MPI_Get_address(  ); ...
for( i=0; i<3; i++ ) tab_odstepow[i] = adres[i] - początek ;
MPI_Type_struct( 3, tab_dlug_blokow, tab_odstepow, tab_typow, &typ1 );
MPI_Type_commit( &typ1 );
```

b) itd., itp. (contiguous, vector, indexed)

Wyjaśnij znaczenie argumentów i funkcjonowanie procedury:

```
int  MPI_Pack( void* buf_dane,   int count,   MPI_Datatype typ,   void* buf_send,   int buf_send_size,
                int* pozycja,   MPI_Comm comm )
```

Jak będzie wyglądało wywołanie procedury MPI_Send dla spakowanych danych?

20. Scharakteryzuj typy systemów komputerowych z pamięcią wspólną. Przedstaw i porównaj typy dynamicznych sieci połączeniowych. Dla każdej sieci zilustruj schematy połączeń procesorów z modułami pamięci i procesorów z procesorami. Porównanie sieci przeprowadź pod kątem: rozszerzalności, wydajności (przepustowości) i kosztu. Omów zasadę trasowania komunikatów w sieci wielostopniowej Ω z przełącznikami 2x2 i idealnym przetasowaniem (uwzględnij rolę dwójkowego zapisu pozycji wejścia i wyjścia). Zilustruj przykładem dla sieci z 8 wejściami i 8 wyjściami.

21. Zilustruj podstawowe typy statycznych sieci połączeniowych. Scharakteryzuj sieć połączeniową w postaci hiperkostki - podaj liczbę procesorów w zależności od wymiaru, sposób kodowania pozycji procesorów, sposób trasowania komunikatów pomiędzy procesorami. Zdefiniuj parametry charakteryzujące statyczne sieci połączeniowe (średnica, połączalność krawędziowa, szerokość połowienia, koszt). Jakie jest znaczenie tych parametrów. Podaj wartości parametrów dla dwóch poniższych sieci (każda mająca p procesorów) i przeprowadź dyskusję ich własności (optymalności) jako narzędzia

przesyłania komunikatów dwupunktowych i grupowych (wybór z: krata 2D, pierścień, torus, hiperkostka 3D itp.).

22. Podaj nazwy (mogą to być nazwy zwyczajowe lub nazwy konkretnych procedur MPI) i zilustruj (obrazy pamięci przed i po operacji dla 3-4 procesorów) schematy grupowego przesyłania komunikatów. Wyjaśnij znaczenie argumentów i funkcjonowanie następującej procedury (jedna z procedur komunikacji grupowej MPI, np.:

```
int MPI_Scatter(void *sbuf,  int scount,  MPI_Datatype sdatatype,  void *rbuf,  int rcount,  MPI_Datatype
    rdatatype,  int root,  MPI_Comm comm)
```

23. Wyjaśnij znaczenie argumentów i funkcjonowanie następującej procedury:

```
int MPI_Reduce(void *sbuf, void *rbuf, int count, MPI_Datatype datatype, MPI_Op op, int root,
    MPI_Comm comm)
```

Jakie istnieją warianty procedury *reduce* w MPI? Jakie istnieją predefiniowane operacje typu MPI_Op i jak można tworzyć własne operacje do użycia przez funkcje *reduce?*

24. Scharakteryzuj model przetwarzania klient-serwer. Przedstaw sposób rozwiązania problemów: wyszukiwania serwera przez klienta oraz przesłania komunikatu pomiędzy klientem i serwerem w ramach mechanizmu gniazd. Przedstaw schemat typowych operacji przy realizacji przetwarzania klient-serwer z transmisją połączeniową. Wyjaśnij znaczenie argumentów i efekt działania następujących funkcji tworzących interfejs gniazd:

a) `int socket(int domain, int type, int protocol);`
b) `int bind(int sockfd,  struct sockaddr *addr,  socklen_t addrlen);`
c) `int listen( int sockfd, int max_dl_kol)`
d) `int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);`
e) `int connect(int  sockfd,  const  struct sockaddr *serv_addr, socklen_t  addrlen);`

Przedstaw i omów program współbieżnego serwera wykorzystującego mechanizm gniazd do połączeniowej komunikacji z klientami. Czym są demony (Unixowe)?

25. Scharakteryzuj model przetwarzania klient-serwer. Przedstaw sposób rozwiązania problemów: wyszukiwania serwera przez klienta oraz przesłania komunikatu pomiędzy klientem i serwerem w ramach mechanizmu gniazd. Przedstaw schemat typowych operacji przy realizacji przetwarzania klient-serwer z transmisją bezpołączeniową. Przedstaw programy klienta i serwera dokonujące prostej wymiany komunikatów w postaci (do wyboru: tablicy znaków, tablicy liczb itp.). Wyjaśnij znaczenie argumentów i efekt działania następujących funkcji tworzących interfejs gniazd:

a) `int socket( int domain, int type, int protocol );`
b) `int bind( int sockfd,  struct sockaddr *addr,  socklen_t addrlen );`
c) `ssize_t  sendto( int  sockfd,  const  void *buf, size_t len, int flags, const, struct sockaddr *to, socklen_t tolen);`
d) `ssize_t recvfrom( int sockfd, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen);`

26. Scharakteryzuj tworzenie programu klienta i programu serwera w modelu zdalnych wywołań procedur Sun RPC. Napisz przykładowy plik interfejsu serwera w języku XDR. Omów mechanizm funkcjonowania wywołań i przesyłania komunikatów w modelu zdalnych wywołań procedur Sun RPC wykorzystujący namiastki po stronie klienta i serwera. Przedstaw sposób rozwiązania problemu wyszukiwania serwera przez klienta w modelu zdalnych wywołań procedur Sun RPC.

27. Omów sposób tworzenia programów CORBA z wykorzystaniem plików IDL, namiastek oraz szkieletów. Przedstaw cechy charakterystyczne języka definiowania interfejsu IDL w ramach specyfikacji CORBA (zwłaszcza różnice w stosunku do C++: moduły, interfejsy, typy, dziedziczenie, definicje (sygnatury) operacji, atrybuty). Co określamy przez semantykę wywołania i jakie są jej typy? Jaką semantykę realizują standardowe operacje zdalne CORBA, a jaką operacje „oneway"? Czym jeszcze operacje „oneway" różnią się od standardowych? Jak funkcjonują wyjątki w środowiskach CORBA?

28. Przedstaw model przetwarzania w systemach CORBA (najlepiej w postaci schematu). Uwzględnij wiązania statyczne (namiastki i szkielety) oraz wiązania dynamiczne. Jaka jest rola rdzenia ORB, a jaka adapterów obiektów? Przedstaw sposób rozwiązania problemów: wyszukiwania serwera (ściślej implementacji obiektu) przez klienta oraz przesłania komunikatu pomiędzy klientem i implementacją obiektu w modelu CORBA. Jakie operacje można wykonywać na dowolnym obiekcie CORBA, ze względu na dziedziczenie po interfejsie Object (wymień lub omów najważniejsze grupy).

29. Przedstaw schemat przetwarzania rozproszonego w architekturze usługowej SOA. W jaki sposób schemat ten jest realizowany przez środowiska WebServices. Przedstaw sposób rozwiązania problemów: wyszukiwania serwera przez klienta oraz przesłania komunikatu pomiędzy klientem i serwerem w modelu WebServices. Omów części składowe pliku opisu usług WSDL. Jaka jest rola XML, SOAP i HTTP w specyfikacjach WebServices?

30. Scharakteryzuj systemy gridowe? Na jakich standardach opiera się ich budowa i kto zarządza tymi standardami? Czym jest wirtualizacja zasobów? Jakie podstawowe zadania realizują systemy gridowe? Jakie znasz systemy gridowe? Scharakteryzuj je krótko.

31. Omów właściwości systemów rozproszonych. Czym charakteryzują się pod kątem: wykorzystywanych zasobów, przezroczystości, niezawodności, wydajności (w tym skalowalności), bezpieczeństwa, otwartości?

32. Scharakteryzuj sposoby synchronizacji czasowej w systemach rozproszonych. Omów algorytmy Cristiana, Berkeley. Czym są znaczniki czasowe i jak za ich pomocą można wprowadzić całkowite uporządkowanie zdarzeń. Omów szczegółowo procedurę.

33. Przedstaw rozwiązania problemu wzajemnego wykluczania w systemach rozproszonych. Omów szczegółowo procedury wykorzystujące: centralny serwer dostępu, pierścień z żetonem i rozproszone uzgadnianie z zastosowaniem zegarów logicznych. Przeanalizuj rozwiązania pod kątem bezpieczeństwa, żywotności oraz wydajności (liczby komunikatów koniecznych do wymienienia w celu realizacji zadania)

34. Zdefiniuj problem elekcji (ogólnie i w konkretnym przypadku do którego dostosowane będą rozwiązania). Przedstaw rozwiązania za pomocą algorytmów: tyrana (Garcii i Molina) oraz stosującego architekturę pierścienia (Changa i Robertsa). Przeanalizuj rozwiązania pod kątem bezpieczeństwa, żywotności oraz wydajności (liczby komunikatów koniecznych do wymienienia w celu realizacji zadania)

35. Scharakteryzuj typy maszyn PRAM ze względu na współbieżność zapisu i odczytu. Napisz i zanalizuj algorytm wyszukiwania konkretnej liczby w n-elementowej tablicy A na n-procesorowej maszynie PRAM (algorytm powinien zawierać komentarze wyjaśniające co dzieje się w kolejnych jego linijkach). Podaj dla jakiego modelu maszyny PRAM jest ten algorytm i jaka jest jego złożoność obliczeniowa? (Wskazówka - nagłowek procedury: wyszukaj(A, p, k, klucz): indeks (lub P/F - prawda albo fałsz))

36. Omów, ilustrując schematami architektur, klasyfikację Flynna komputerów równoległych. Jaki jest związek tej klasyfikacji ze współczesnym podziałem systemów równoległych ze względu na dostęp do pamięci. Scharakteryzuj współczesne typy komputerów równoległych (m. in. wyjaśnij skróty i omów opisywane nimi architektury: UMA, ccNUMA, SMP, DSM, MPP).

37. Omów sposoby tworzenia programów równoległych. Przedstaw rodzaje dekompozycji zadania obliczeniowego.

38. Podaj definicję przyspieszenia obliczeń równoległych i charakter typowego wykresu przyspieszenia jako funkcji liczby procesorów. Podaj definicję efektywności zrównoleglenia obliczeń i charakter typowego wykresu efektywności jako funkcji liczby procesorów. Wyjaśnij pojęcia (symbole) pojawiające się w definicjach (wzorach).

39. Przedstaw analizę Amdahla i prawo Amdahla jako jej konsekwencję w przypadku rosnącej liczby procesorów (wyprowadź wzory i zilustruj przykładami wykresów czasu wykonania, przyspieszenia i efektywności). Przedstaw analizę Gustafsona (analiza wykonania równoległego przy stałym czasie). Jakie są wnioski z obu tych analiz? Czym jest skalowalność obliczeń?

40. Omów składniki czasu wykonania rzeczywistych programów równoległych (z czym związany jest dany składnik, w jaki sposób można oszacować jego wielkość). Przedstaw

ilustrację czasu wykonania dla przykładowego zadania i kilku procesorów. Jakie są czynniki obniżające wydajność obliczeń równoległych?

41. Scharakteryzuj model programowania z równoległością danych (*data parallel programming*). Podaj operacje (instrukcje) języka HPF, które dopuszczają realizację równoległą. Omów je krótko. Przedstaw i zilustruj kilkoma przykładami sposób definiowania układu procesorów w HPF. Omów konstrukcje języka High Performance Fortran służące do określenia rozłożenia danych (wektorów i macierzy) na procesory. Zilustruj przykładami. Naszkicuj schemat przyporządkowania poszczególnych wyrazów tablic poszczególnym procesorom w przypadku następujących deklaracji i dyrektyw HPF:

a)

```
!HPF$   PROCESSORS, DIMENSION(...,...):: P
REAL, DIMENSION(...,...):: T
!HPF$   DISTRIBUTE   T(...,...)   ONTO   P
```

b)

```
!HPF$ PROCESSORS, DIMENSION(...,...):: P
REAL, DIMENSION(...,...):: T1
REAL, DIMENSION(...,...):: T2
!HPF$ DISTRIBUTE   T1(...,...)   ONTO   P
!HPF$ ALIGN T2(...,...) WITH T1(...,...)
itd., itp.
```