Systemy wieloprocesorowe

Sprzęt i oprogramowanie wspomagające perspektywa - Windows i Linux

Wykład Przetwarzanie równoległe

Listopad 2010

Pożądane cechy aplikacji wielowątkowych

Skalowalna wielowątkowość przetwarzania przez:

- podział na zadania: funkcjonalny i domenowy - oparty o dane,
- organizacja danych w niezależne zbiory, powielenie danych czytanych.
- efektywne korzystanie z pp:
  - prywatne dane wątków
  - zapobieganie niezamierzonemu współdzieleniu (np. wyrównanie 64 bajtowe przy alokacji danych)
- przydatna znajomość sprzętu dla optymalizacji kodu

NUMA versus SMP
systemy wieloprocesorowe

NUMA- każdy procesor jest bliżej pewnych części pamięci niż innych (nie dotyczy pamięci podręcznej). W systemach NUMA system operacyjny próbuje szeregować wątki na te procesory, które są bliżej pamięci, którą wykorzystują .

SMP – procesory i rdzenie podłączone są do pojedynczej współdzielonej pamięci głównej, przepustowość magistrali łączącej procesory z pamięcią stanowi silne ograniczenie dla efektywności systemu przy zwiększającej się liczbie procesorów. W SMP wątek może być przydzielony do dowolnego procesora. Program szeregujący dysponuje zbiorem procesorów, które są wykorzystywane do współbieżnego przetwarzania wątków. Szeregowanie jest uzależnione od priorytetów wątków, lecz optymalizacja wykorzystania pamięci podręcznej w procesach wielowątkowych może być realizowana za pomocą funkcji ustalających: **Thread Affinity – powinowactwo wątków i Thread Ideal Processor – najlepszy procesor dla wątku**

Powinowactwo wątków – SMP
(Windows)

- Funkcje wymuszają uruchomienie wątku na specyficznym podzbiorze procesorów.
- Ustalenie powinowactwa wątków może kolidować z regułami szeregowania i może utrudniać modułowi szeregującemu uzyskanie efektywności przetwarzania w całym systemie.
- System reprezentuje powinowactwo za pomocą maski bitowej – maski powinowactwa wątków, rozmiar maski jest równy maksymalnej liczbie procesorów w systemie, a ustawione bity (=1) określają podzbiór procesorów. System określa pierwotny stan maski.
- Funkcje: **GetProcessAffinityMask** **SetProcessAffinityMask** **SetThreadAffinityMask**

Funkcje powinowactwa

Windows

- **GetProcessAffinityMask** - uzyskanie aktualnej maski powinowactwa dla wszystkich wątków danego procesu.
- **SetProcessAffinityMask** - ustalenie aktualnej maski powinowactwa dla wszystkich wątków danego procesu.
- **SetThreadAffinityMask** - ustalenie aktualnej maski powinowactwa dla pojedynczego wątku. Powinowactwo wątków musi być podzbiorem powinowactwa procesu.
- Dla systemów z więcej niż 64 procesorami maska powinowactwa określana jest dla pojedynczej grupy procesorów.

Funkcje pomocnicze (szeregowanie)

Windows

- SetThreadIdealProcessor określa preferowany procesor dla wątku lecz nie gwarantuje jego wyboru, jest informacją pomocniczą dla modułu szeregującego.
- GetLogicalProcessorInformation dostarcza informacji o procesorach logicznych w systemie i zależnościach między nimi (np. współdzielony rdzeń, procesor, węzeł NUMA) (dla aktualnej grupy procesorów)

Wspomaganie przydziału procesów/wątków i pamięci (Linux)

Rozszerzenia GNU deklarowane w sched.h

Typ danych: **cpu_set_t** - pozwala okreslić maskę powinowactwa (ustalić gdzie uruchomic wątek) każdy bit odpowiada innemu CPU

Makro int **CPU_SETSIZE** - dostarcza liczby procesorów obsługiwanych przez cpu_set_t

Makra do manipulacji na typie cpu_set_t

void **CPU_ZERO** (*cpu_set_t *set*)
void **CPU_SET** (*int cpu, cpu_set_t *set*)
CPU_CLR (*int cpu, cpu_set_t *set*)
int **CPU_ISSET** (*int cpu, const cpu_set_t *set*)

Wspomaganie przydziału procesów/wątków i pamięci (Linux)

- Funkcje do poznania i określenia maski powinowactwa dla procesów i wątków:

  int **sched_getaffinity** (*pid_t pid, size_t cpusetsize, cpu_set_t *cpuset*)

  int **sched_setaffinity** (*pid_t pid, size_t cpusetsize, const cpu_set_t *cpuset*

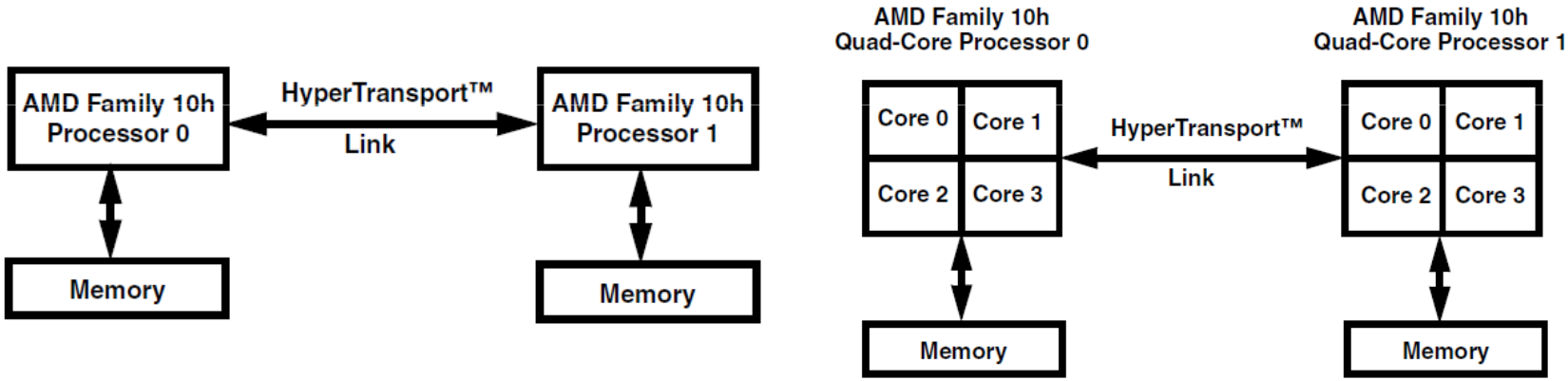
Dodatkowe narzędzia dla SMP (np. Intel Xeon 5300) do określania pozycji w masce powinowactwa poszczególnych rdzeni dostępnych w systemie przynależnych lub nie do jednego procesora, współdzielących (np. parami) pamięci podręczne L2 lub L3.

Systemy NUMA

Systemy NUMA pozwalają na wzrost prędkości przetwarzania bez konieczności wzrostu obciążenia magistrali procesora. Procesory uzyskują szybciej dostęp do bliskiej im pamięci, a do dalszej czas dostępu może być dłuższy. Jednostki przetwarzające tworzą podsystemy zwane węzłami. Każdy węzeł posiada własne procesory i pamięć, węzły są połączone z innymi węzłami za pomocą magistral zapewniających spójność pamięci podręcznych.

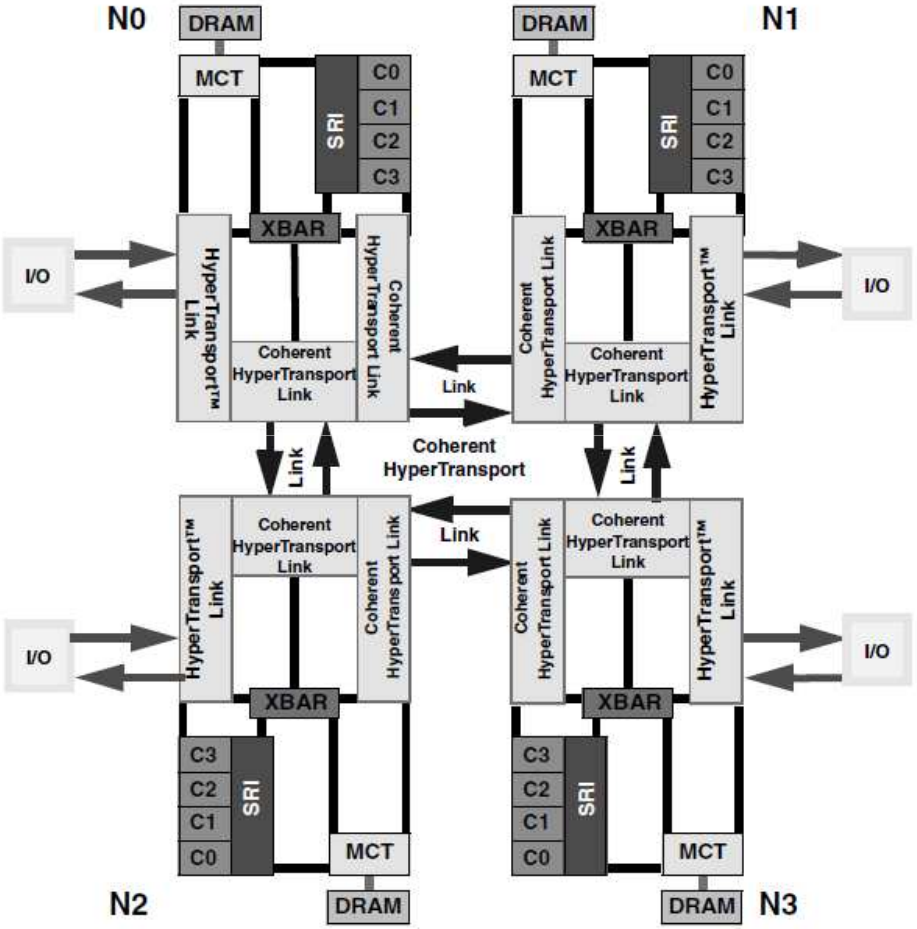
System operacyjny próbuje zwiększyć wydajność przez szeregowanie wątków do procesorów w węzłach, gdzie znajduje się wykorzystywana przez nie pamięć, próbuje realizować żądania przydziału pamięci w ramach tego samego węzła, lecz również innych jeśli będzie to konieczne. System dostarcza funkcji pozwalających na określenie topologii systemu dostępnego dla aplikacji. Efektywność aplikacji może zostać podwyższona za pomocą funkcji dla NUMA pozwalających na optymalizację szeregowania i wykorzystania pamięci.

Systemy NUMA przykład I



System dwuprocesorowy rodziny AMD 10h

SYSTEMY NUMA PRZYKŁAD II



Cztery rdzenie komunikują się przez: **system request interface** (SRQ) połączony z **non-blocking crossbar** (XBar). **XBar** jest połączony ze sterownikiem pamięci (**MCT**) i różnymi łączami typu HyperTransport . Sterownik pamięci MCT jest połączony z pamięcią lokalnego węzła. MCT, SRQ i XBar każdego węzła posiadają wewnętrzne bufory wykorzystywane do kolejkowania pakietów transakcji HyperTransport . SRQ, XBar i MCT tworzą Northbridge węzła.

System 4 procesorów 4 rdzeniowych rodziny 10h AMD

Szeregowanie wielu wątków w systemie wieloprocesorowym NUMA

- Dane współdzielone – kolejne wątki należy przydzielać do różnych rdzeni tego samego procesora
- Dane niezależne – kolejne wątki należy przydzielać do różnych procesorów

ZAGADNIENIA LOKALNOŚCI DANYCH - NUMA

- Realizować w kodzie dostęp do danych węzła przez wątek będący lokalnym dla tego węzła – W przypadku danych niezależnych każdy wątek powinien alokować pamięć i zainicjować (zapis) dane, które będzie wykorzystywał - pozwala to na podjęcie przez system operacyjny właściwych decyzji odnoście lokalizacji danych (w pamięci węzłów).
- Wywłaszczenie wątku powoduje niebezpieczeństwo jego powrotu na inny węzeł i inicjowania tam kolejnych danych co daje rozproszony fizycznie system danych. Warto zatem definiować powinowactwo (affinity) wątków, gdyż ułatwia to efektywne szeregowanie wątków.

LOKALNOŚĆ DANYCH NUMA

Zalecane podejścia:

- Dane współdzielone – warto dane współdzielone przez wiele wątków rozproszyć przeplot lokacji stron danych aplikacji, aby dostęp do nich mógł być zrównoleglony , a korzystanie z pamięci nie było wąskim gardłem.
- Usunięcie niekoniecznego współdzielenia lini pp między wątkami (64 bajty linii pp) - usunięcie niezamierzonego współdzielenia (ang. false sharing)
- Stosowanie zamków korzystających z pamięci podręcznej (minimalizacja ilości blokad na magistralach między węzłami – zapobieganie serializacji) (cachable locks)

NUMA – STRUKTURA SYSTEMU

(Windows)

Określenie struktury systemu jest możliwe za pomocą nastepujących funkcji:

- GetNumaHighestNodeNumber – dostarcza najwyższy numer węzła - nie jest to koniecznie liczba węzłów, sąsiednie numery nie muszą dotyczyć sąsiednich w architekturze węzłów
- GetProcessAffinityMask – dostarcza liczby procesorów w systemie
- GetNumaProcessorNode – określa węzeł dla danego procesora
- GetNumaNodeProcessorMask – określa listę procesorów danego węzła
- Dodatkowo użyteczne są także w systemie NUMA: **GetProcessAffinityMask** , **SetProcessAffinityMask, SetThreadAffinityMask** oraz **GetLogicalProcessorInformation**

FUNKCJE PRZYDZIAŁU PAMIĘCI - NUMA
(Windows)

- VirtualAlloc – rezerwuje stronę pamięci wirtualnej lokowaną fizycznie na lokalnym węźle NUMA (na tym na którym nastąpi pierwszy dostęp do tej strony lub preferowanym przez system) (zapobieganie rozproszeniu danych między węzłami)
- VirtualAllocExNuma - rezerwuje stronę pamięci wirtualnej lokowaną fizycznie na preferowanym węźle NUMA
- VirtualFree, VirtualFreeEx - zwalniają pamięć

WSPOMAGANIE PRZYDZIAŁU PROCESÓW/WĄTKÓW I PAMIĘCI - NUMA (Linux)

- *cat /proc/cpuinfo* - listing dostępnych procesorów w systemie
- int NCPUs = sysconf(_SC_NPROCESSORS_CONF); zwraca liczbę procesorów dostępnych dla systemu operacyjnego (#include <unistd.h>)
- **Interfejs programistyczny NUMA API** (do sterowania powinowactwem wątków i pamięci) składa się z:
  - części jądra zarządzającej przydziałem pamięci (ważne w NUMA, gdyż w odróżnieniu od pamięci podręcznej gdzie dane idą za wątkiem przydzielonym do procesora, w przypadku ulokowania danych w pamięci rozproszonej raz przydzielone do określonej lokacji dane wracają zawsze do niej),
  - biblioteki przestrzeni użytkownika ***libnuma*** dołączanej do aplikacji i
  - wywołania linii zleceń: ***numactl***
    - pozwala na zarządzanie powinowactwem wątków i procesów, sterowaniem przydziału pamięci w NUMA (przeplot stron między węzłami)
    - numactl --cpubind=0--membind=0,1 proces –
      - uruchamia proces na węźle 0 z pamięcią przydzieloną do węzłów 0 i 1.
    - numactl –hardware – dostarcza informacji o dostępnych węzłach w systemie

SYSTEMY DLA LABORATORIUM PR I

- Komputery znajdujące się w ***Laboratorium Systemów Równoległych*** posiadają po jednym procesorze AMD typu PHENOM II X4 945
- System składa się z 4 procesorów logicznych – 4 rdzeni w ramach jednego procesora. System SMP.

PROCESOR PHENOM

Zgodność 32 bitowa X86 IA

wspomaganie SSE, SSE2, SSE3, SSE4a, ABM, MMX™, 3DNow!™

Streaming **S**IMD **E**xtensions

Technologia AMD64

rozszerzenia AMD64 technology instruction-set

Adresowanie 48-bitowe

16 rejestrów 64-bit dla integer

16 rejestrów 128-bit SSE/SSE2/SSE3/SSE4a

Architektura wielordzeniowa

Triple-core, **quad-core** lub six-core opcje

AMD Balanced Smart Cache

oddzielne pp L1 i L2 dla każdego rdzenia

współdzielona L3

Struktura procesora

supersklarny 3 drożny (dekodowanie, wykonanie integer i FP, generacja adresu)

Struktura pp

64-Kbyte 2 drożna dzielona asocjacyjna pp danych L1

dwa dostępy 64-bit na cykl, 3 cyklowe opóźnienie

64-Kbyte 2 drożna dzielona asocjacyjna pp kodu L1

32 bajtowe pobrania

512-Kbyte 16 drożna dzielona asocjacyjna pp L2

Zarządzanie pamięcią na zasadzie wyłączności przechowywania danych L1 i L2

6-Mbyte Maximum, maksymalnie 64 drożna dzielona asocjacyjna pp L3 współdzielona

Technologia 45 nm

Złącze HyperTransport™

Procesor zintegrowany ze sterownikiem pamięci

PROCESOR PHENOM – PP KODU L1

- Układ dynamicznego wykonania instrukcji posiada 64KB pp kodu L1
- Dane-kod w przypadku braku trafienia są pobierane do pp kodu L1 z L2, z L3 lub z pamięci systemowej w ilości 64 bajtów (pobranie) oraz kolejne 64 bajty (wstępne pobranie), po pobraniu realizowany jest wstępne dekodowanie instrukcji dla określenia granic między instrukcjami (zmiennej długości), usuwanie linii z pp jest realizowane zgodnie z algorytmem LRU (ang. least recently used**)**

PROCESOR PHENOM – PP DANYCH L1

- 64 kB dwu-sekcyjna, dwa porty 128 bitowe
- Write-allocate cache – zapis realizowany do pp (przeciwna do nowrite allocation)
- Writeback cache – zapis poza pp realizowany w przypadku braku miejsca lub na skutek zlecenia zapisu stanu w pamięci głównej
- Algorytm LRU i protokół CC MOESI

PROCESOR PHENOM – PP L2 I L3

- PP L2 - victim i copy-back cache – zapisuje dane usunięte z pp L1, dane w pp są w trybie wyłącznym w L1 lub w L2
- PP L3 – victim i copy-back cache dla pp L2, głównie non-inclusive cache w przypadku, gdy dane żądane są przez jeden z rdzeni i jest mało prawdopodobne, że będą potrzebne innym, możliwe powielenie.

INTEL NUMA –
Nehalem Intel Microarchitecture

- Procesor Xeon serii 3500 i 5500 (technologia 45 nm)
- Intel® Hyper-Threading Technology – współdzielenie przez wiele wątków zasobów jednego rdzenia procesora – efekt: jednoczesna realizacja 2 wątków na rdzeń
- Intel® QuickPath Technology – nowa skalowalna, architektura współdzielonej pamięci, kontroler pamięci zintegrowany z procesorem, połączenie procesorów i innych komponentów nowym systemem łączy nowego typu z prędkością do 32 GB/s na łącze dwukierunkowe (4 łacza)
- Możliwość przetwarzania do 4 instrukcji na cykl,
  - Wzrost okna instrukcji szeregowania dynamicznego – efektem większy potencjalny poziom równoległości
  - Szybsze mechanizmy synchronizacji wątków
  - Szybsza obsługa złej predykcji rozgałęzień kodu
- Dynamiczne zarządzanie rdzeniami, wątkami, pp, łączami i mocą
  - Dynamiczne zarządzanie rdzeniami – możliwość zwiększenia częstotliwości zegara wszystkich rdzeni w przypadku potrzeby/ dodatkowy wzrost f dla grupy aktywnych rdzeni, możliwe różne zegary rdzeni (nie jak w Core 2)

INTEL NUMA –

Nehalem Intel Microarchitecture

- Współdzielona (przez wszystkie rdzenie procesora jak pp L2 w Intel Core) pamięć podręczna 3 poziomu do 8 MB (powiela dane zapisane w innych poziomach pp) – efekt: mniejszy ruch pamięć – rdzeń (pp L1 L2) – powielenie powoduje spadek pojemności systemu pamięci, lecz umożliwia zmniejszenie ilości dostępów do L1 i L2 (brak konieczności przeszukiwania L1 i L2 przy cache miss do L3, zmniejszenie narzutu przeszukiwania (którym rdzeniu dane powielone?) przy cache hit – dzięki dodatkowym flagom w L3)
- 1 do 8 rdzeni, 1 do 16 wątków
- pp L1 (32KB IC + 32KB DC)
- pp L2 256 KB na rdzeń
- 2 poziomy struktury TLB (Translation Lookaside Buffer) – translacja adresów pamięci wirtualnych - systemu na fizyczne – sprzętu (192 + 512 wejść)

Przykład: Optymalizacja transmisji danych między rdzeniami

(producent-konsument)

Systemy z współdzieloną między rdzeniami pp L3 (wyłączna).

Wątek uruchomiony na jednym rdzeniu produkuje dane konsumowane przez inny wątek na sąsiednim rdzeniu tego samego procesora (efektywna wymiana danych bezpośrednio przez pp L3).

pp write-back - dane usuwane z L1 trafiają do L2, zaś usuwane z L2 do L3.

Określenie odległości w buforze pomiędzy danymi dostarczanymi, a pobieranymi.

- odległość jest większa od sumy rozmiarów L1 i L2 (dlaczego?)
  - aby konsument znalazł interesujące go dane w pp L3.
- odległość ta nie może być jednoczesnie zbyt duża > L3+L2+L1,
  - aby nie spowodować usunięcia z pp L3 danych nieskonsumowanych jeszcze przez konsumenta.

Określenie odległości w buforze pomiędzy obszarem wykorzystywanym, a dostepnym w L3

- odległość jest większa od sumy rozmiarów L1 i L2 (dlaczego?)
  - Aby producent znalazł interesujący go obszar bufora z powrotem w pp L3 (w odpowiednim stanie CC protokół MOESI).

SYNCHRONIZACJA W WINDOWS

Wykorzystuje: zdarzenia (events), zamki(mutex), semafory i liczniki

- Zamki - CreateMutex, OpenMutex, ReleaseMutex, WaitForSingleObject, WaitForMultipleObjects (funkcje oczekiwania)
- Zdarzenia - CreateEvent, SetEvent, ResetEvent, funkcje oczekiwania
- Semafory - CreateSemaphore, ReleaseSemaphore, OpenSemaphore, funkcje oczekiwania
- Liczniki - CreateWaitableTimer, SetWaitableTimer, funkcje oczekiwania

Literatura

http://msdn.microsoft.com – opis bibliotek systemów Windows

WHITE PAPER Intel® Xeon® processor 3500 and 5500 series Intel® Microarchitecture

Family 10h AMD Phenom™ II Processor Product Data Sheet

Software Optimization Guide for AMD Family 10h Processors

NUMA optimization in Windows Applications Michael Wall, AMD