|  |  |  |
| --- | --- | --- |
| Slajd 1 |  | Koprocesory architektury Intel MIC wykorzystywane do równoległych obliczeń dużej skali. |
| Slajd 2 |  |  |
| Slajd 3 |  | Są to po prostu dodatkowe karty PCI Express, które współdziałają z procesorami Intel Xeon znacznie zwiększając wydajność przetwarzania równoległego w dużej skali.  Tranzystory Tri-Gate był pierwszymi tranzystorami 3D produkowanymi masowo. Bariery fizyczne przy coraz większej miniaturyzacji ograniczały dalszy rozwój, dlatego od 2011 Intel wprowadza tą technologie. Lepsza kontrola gate’a, łatwiej wyłączać. W 2012 wprowadzone komercyjnie. |
| Slajd 4 |  | Przedewszystkim koprocesory Xeon mają więcej rdzeni i wątków. VPU dużo większe, jedyną różnicą in minus jest częstotliwość taktowania. Ogólnie przekłada się na większą łączną wydajność o czym się dziś przekonamy. |
| Slajd 5 |  | Intel Xeon zapewnie odpowiednią wydajność dla pewnie 90% obliczeniowych aplikacji. Jednak tam gdzie mamy doczynienia z aplikacjami równoległymi dużej skali koprocesory Xeon Phi znacznie poprawiają wydajność pracy. Szczególnie służą tam, gdzie wykorzystuje się ponad 100 wątków oprogramowania oraz obliczenia na wektorach lub tam gdzie potrzebna jest większa przepustowość pamięci lokalnej niż ta oferowana przez Intel Xeon |
| Slajd 6 |  |  |
| Slajd 7 |  |  |
| Slajd 8 |  | wszędzie gdzie potrzebne są wielkoskalowe obliczenia równoległe  animacja, energetyka, finanse, nauki przyrodnicze, produkcja, medycyna, prognozowanie pogody  Na końcu prezentacji przedstawimy dwa bardzo konkretne zastosowania, na bazie tego co powiemy o tym produkcie. |
| Slajd 9 |  | Koprocesory Phi obsługują do 61 rdzeni, każdy po cztery fizycznie wątki, tak więc 244 wątki. Taktowanie od 1 do 1.3 GHz. Architektura bazuje na x86 Instruction Set Architecture (ISA – to wszystko co w architekturze związane jest z programowaniem: natywne typy danych, instrukcje koprocesora, rejestry, sposoby adresowania, budowa pamięci, obłsuga wyjątków przerwań, I/O).  Instrukcje SIMD są wykonywane na 512 bitowych rejestrach na VPU (Vector Processing Unit)  Poza rdzeniami mamy kontrolery pamięci, magistrale i trochę dodatkowych komponentów. |
| Slajd 10 |  | Tutaj mamy porównanie z seriami procesorów Intel Xeon. Xeon Phi ma ponad 2x większą przepustowość lokalnej pamięci niż rodzina E5, jest 4x oszczędniejszy jeśli chodzi o pobór energii. |
| Slajd 11 |  | Tutaj widzimy generalny widok architektury MIC. Rdzenie ułożone są w postaci pierścienia. Wszystkie L2 cache są ze sobą połączone w jedną wspólną pamięć. Mamy dostępną szybką pamięć GDDR5 o 4KB domyślnego rozmiaru strony. Rdzenie bazowane są na starym procesorze Pentium (P65C) – główne modyfikacje to dodanie adresowania 64-bit, SIMD, wspólne L2 cache.  Pamięć GDDR5 jest 8GB z ośmioma kontrolerami pamięci, 16 kanałami. Dostęp do niej zajmuje 300ns. W sumie przepustowość może wynieść 352GB/s. |
| Slajd 12 |  | Na diagramie blokowym widzimy budowę: mamy dekoder instrukcji, jednostkę skalarną i wektorową ze swoimi rejestrami, cache L1 (do instrukcji i do danych) i L2. Wszystko tutaj jest dokładnie takie samo jak to czego się uczyliśmy, Intel celowo zaprojektował architekturę, z którą już wszyscy mieli do czynienia.  Tak więc nie trzeba się uczyć nowych paradygmatów: zarówno na poziomie softwarowym jak i na poziomie hardwarowym mamy doczynienia z tym samym co już znamy.  Pierwsza różnica jaką zauważymy to fakt, że mamy doczynienia z ponad 60 rdzeniami połączonymi razem za pomocą ring interconnectora (rys. z poprzedniego slajdu). Używa się scalar unitów z generacji Pentium – wolniejsze oraz SIMD Vector Processing Engine  Każdy rdzeń ma 32KB L1 cache na dane i na instrukcje oraz 512KB L2 cache, cache L2 są połączone ze sobą nawzajem i z kontrolerami pamięci przez dwukierunkową magistrale w kształcie pierścienia, co w rezultacie daje pamięć współdzieloną 32 MB. Mamy ponadto krótki potok przetwarzania.  ISA: 512-bit SIMD operacje na wektorach – 8 operacji na doublach na jeden takt zegara. Maski wektorowe, które ustawiają, które elementy wektora będą potraktowane przez instrukcje, obecny EMU (Extended Math Unit). Większość instrukcji wektorowych mają opóźnienie 4 taktów zegara. |
| Slajd 13 |  | To tyle jeżeli chodzi o hardware. Teraz spójrzmy jak wygląda architektura software dla tego koprocesora. Dzielimy tutaj system na linuksowego hosta i architekturę softwarową koprocesora. Na poziomie kodu systemowego zarówno od strony hosta jak i koprocesora działamy na linuksie. Koprocesor działa na SMP-on-a-chip Linux czyli na systemie wieloprocesorowym ze współdzieloną pamięcią. Fizyczna komunikacja jest dostępna poprzez interfejs PCIe. Software po obu stronach różni się niewiele. Od strony koprocesora nie ma tylu bibliotek co po stronie hosta. Alternatywnym sposobem uruchamiania aplikacji jest kompilowanie po stronie hosta i wysyłanie przy użyciu protokołu SSH do koprocesora. Zazwyczaj platformę sprzętową buduje się tak, że mamy 1 lub 2 procesory Xeon jako hosty i do nich dołączone 1-8 koprocesorów Xeon Phi przypadające na hosta. Każda karta ma swój adres IP. I tak używając terminala na hoście i wpisując polecenie ssh mic0 możemy połączyć się z pierwszym koprocesorem na naszej platformie. |
| Slajd 14 |  | Będąc zalogowanym możemy wyświetlić informacje na temat koprocesora. Informacja ta zajmuje ponad 6000 linii. Tutaj pokazujemy pierwsze i ostatnie linie. |
| Slajd 15 |  |  |
| Slajd 16 |  | Oczywiście nie ma sensu pisać równoległej aplikacji gdy sprzęt tego nie wspiera, i odwrotnie mamy super równoległy sprzęt kilkadziesiąt fizycznych wątków, ale program nie jest dobrze napisany. |
| Slajd 17 |  |  |
| Slajd 18 |  | Można więc postawić wniosek: jeśli chcesz by program twój działał szybciej potrzebne jest większe zrównoleglenie operacji, a w przypadku Xeon Phi jeżeli chcesz uzyskać ten sam performance jak na Xeon musisz również bardziej zrównoleglić (overhead na obsługę) |
| Slajd 19 |  |  |
| Slajd 20 |  |  |
| Slajd 21 |  | W trybie natywnym:   * karta rozszerzeń Phi jest po porstu oddzielnym SMP (architektura umożliwiająca przetwarzanie równoległe) z Linuxem * Oddzielne programy wykonawcze na koprocesorze i procesorze Xeon * Praktycznie żadnej modyfikacji kodu źródłowego, wystarczy zrekompilować na koprocesorze * Działa jako ACN – Autonomous Compute Mode   W trybie offload:   * Main działa na hoście * Fragmenty kodu są offloaded na MIC * Kod który się przenosi musi być wielowątkowy, wysoko zrównoleglony, wektoroable, korzystać z dużego BW lokalnej pamięci |
| Slajd 22 |  |  |
| Slajd 23 |  |  |
| Slajd 24 |  | Koprocesor może pracować w wielu trybach wykonawczych:   * Tryb Host Native- jedynie host zarządza zadaniami (wtedy gdy nie da się zrównoleglić programu) * Offload – tryb odciążenia: centralny procesor zarządza zadaniami, w razie potrzeby część przejmowana jest przez koprocesor * Tryb Symetryczny – przetwarzane zadania dzielimy pomiędzy procesor centralny i koprocesor * Reverse offload – to odwrócony tryb odciążenia, gdzie zarządzanie odbywa się po stronie koprocesora, a część zadań przejmowana jest przez hosta * MIC Native – tryb natywny, najbardziej MIC-centric: koprocesor całkowicie zarządza zadaniami, działa jak oddzielne węzły obliczeniowe |
| Slajd 25 |  |  |
| Slajd 26 |  | Przedstawiony na slajdzie kod jest najprostszym możliwym sposobem jak stworzyć wątki z pętli for przy użyciu dyrektywy kompilatora pragma w OpenMP. OpenMP to wieloplatformowe API umożliwiające tworzenie programó komputerowych dla systemów wieloprocesorowych z pamięcią dzieloną. Wykorzystywane w C/C++ i Fortranie na wielu architekturach: Unix, Windows. Jest to przenośny całkowicie skalowalny standard zatwierdzony przez głównych producentów sprzętu. |
| Slajd 27 |  |  |
| Slajd 28 |  |  |
|  |  |  |