Intel Xeon Phi

Raport z prezentacji

1. Wstęp

Tematem referatu była rodzina koprocesorów Intel Xeon Phi. W prezentacji poruszyliśmy tematy związane z architekturą MIC, sposobami programowania koprocesora, jego zastosowaniem.

Przedstawione zagadnienia to:

- generalna filozofia rozwiązania
- architektura sprzętowa oraz systemowa urządzenia
- metodyka programowania koprocesora
- przykłady zastosowań w nauce i przemyśle

2. Streszczenie prezentacji

Wprowadzenie do Intel Xeon Phi.

Intel Xeon Phi to koprocesory architektury Intel MIC wykorzystywane do równoległych obliczeń dużej skali. Są to dodatkowe karty PCI Express, które współdziałają z procesorami Intel Xeon znacznie zwiększając wydajność przetwarzania równoległego w dużej skali.

Zależność pomiędzy procesorem Intel Xeon a koprocesorem Phi.

Koprocesory Xeon mają więcej rdzeni i wątków. Posiadają dużo większe jednostki wektorowe, mając przy tym niższą częstotliwość taktowania. W rezultacie cechują się większą wydajnością obliczeniową przy spełnieniu kilku warunków.

Filozofia stosowania

Intel Xeon zapewnia odpowiednią wydajność dla 90% 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.

Architektura sprzętowa, Many Integrated Core.

Koprocesory Phi obsługują do 61 rdzeni, każdy po cztery wątki. Taktowanie zegara to 1.3 MHz (bazują na rdzeniu Pentium P65C). Architektura bazuje na x86 ISA z dodanym adresowanie 64-bitowym i 512-bitowymi jednostkami SIMD. Xeon Phi ma ponad 2 razy większą przepustowość pamięci lokalnej w porównaniu do rodziny procesorów E5 i jest od nich 4 razy oszczędniejszy energetycznie.

Rdzenie ułożone są w pierścień, połączone przez cache L2 w jedną wspólną pamięć przy użyciu dwukierunkowej magistrali. Wyposażone są w pamięć GDDR5 8 GB. Umożliwione jest w nich przetwarzanie potokowe. Za dekoderem instrukcji dzielą się na jednostkę skalarną i wektorową. Cache L1 podzielona na cache instrukcji i danych.

Rdzenie wykonane są w technologi tranzystorowej Tri-Gate. Były pierwszymi tranzystorami 3D produkowanymi masowo.

Architektura oprogramowania - pełnoprawny system Linux po stronie koprocesora.

Wydzielamy software działający na hoście i koprocesorze. Na obu jednostkach jest to Linux (SNMP-on-a-chip). Każda karta ma swój adres IP, występuje pełna obsługa sieciowa.

Porównanie wyników bechamarków z zależności od liczby wątków.

Dopiero odpowiednie zrównoleglenie operacji daję przewagę nad "zwykłymi" procesorami Xeon. To samo jeśli chodzi o wykorzystanie pamięci lokalnej.

Tryby wykonawcze: tryb native, tryb offload, tryb symmetric

W trybie natywnym karta rozszerzeń działa jako oddzielny SMP. Dostarczamy oddzielne programy wykonawcze na procesor i koprocesor Phi. Brak przy tym potrzeby modyfikacji kodu źródłowego (wystarczy rekompilacja).

W trybie offload mamy główną logikę programu na hoście (procesorze), a wykonanie oznaczonych fragrmentów kodu jest przenoszone na koprocesor. Kod, który się przenosi musi być wielowątkowy, korzystać z VPU i wysokiego BW pamięci lokalnej.

Tryb symetryczny to podwójny tryb natywny: specjalne autonomiczne programy na hoście i koprocesorze. Tryb reverse offload to odwrotność trybu offload tzn. głowna logika na koprocesorze, a fragmenty wykonania na hoście.

Przykłady kodu z zastosowaniem OpenMP

```
int main (int argc, char *argv[])
{
	int nthreads, tid;
	/* Fork a team of threads giving them their own copies of variables */

	#pragma offload target (mic)
	{
		#pragma omp parallel private(nthreads,tid)
		{
		/* Obtain thread number */
		tid = omp_get_thread_num();
		printf("Hello World from thread = %d\n", tid);

		/* Only master thread does this */
		if (tid == 0) {
			nthreads = omp_get_num_threads();
			printf("Number of threads = %d\n", nthreads);
		}
			#ifdef __MIC__
				printf("on target...\n");
			#else
			printf("on host...\n");
			#endif
		}
	}
}
```

Modele transferu danych: implicit, explicit.

Można explicite oznaczać zmienne do offloadowania na koprocesor. Zostaną utworzone wtedy kopie w pamięci koprocesora (oddzielne od hosta). Możemy również oznaczać zmienne, które mają być dzielone pomiędzy hosta i koprocesor. Wtedy również fizycznie występuje kopiowanie zawartości tych zmiennych, ale runtime utrzymuje ich spójność tak by wyglądało to na współdzielenie.

Optymalizacja użytkownika

Przedstawiliśmy tutaj kilka sposób na wykonanie optymalizacji: transformacje pętli, dostępu do pamięci, zagęszczanie wektorów, uspójnianie pamięci. Optymalizacja użycia wektorów, użycie dużych stron, dobór algorytmów umożliwiających lepsze zrównoleglenie aplikacji.

Model kompilacji i programowania

Wytłumaczyliśmy tu po krótce jak kompiluje się rogram napisany pod koprocesor Xeon Phi. Jakich metod, narzędzi i bibliotek się używa: Intel Math Kernel Library, API w C, C++ i Fortranie (OpenMP, ITBB, MPI). Najpopularniejsze z nich przedstawiliśmy z przykładami.

Przykłady rozwiązań: Cosmos, NEC Superresolution Processing

Dwa praktyczne zastosowania, gdzie stosunkowo niewielkim nakładem pracy osiągniętą znaczącą poprawę wydajności obliczeń. Przedstawione projekty to komputer COSMOS, na którym symuluje się ewolucje ścian domenowych w początkach Wszechświata. Zmiana kodu na cache-friendly, wektoryzacja i zmniejszenie zajętości pamięci przy zastosowaniu prostego OpenMP pozwoliło osiągnąć 17 razy lepsze wyniki w stosunku do tych przed zmianą. Drugim przykładem była implementacja konwersji SD->HD na serwerach NEC wyposażonych w Xeon Phi. Przed migracją film 30fps konwertowany był z szybkością 6fps. Po migracji ten sam film udało się konwertować w real-time.

3. Wnioski

Przy rozważaniu przeniesienia obliczeń na platforme Intel Xeon Phi trzeba rozważyć wiele aspektów, z których najważniejszym jest trudność jaka będzie związana z dostosowania naszego kodu do możliwości karty. Warto też zapamiętać wymienione metody zrównoleglania kodu i jego optymalizacji pod względem wielowątkowości. Na uwagę zasługuje również prostota trybów wykonawczych i przenoszenia kodu z platformy Xeon i przewaga koprocesorów nad GPU w wielu zastosowaniach. Do przenoszenia projektów powinna zachęcać prostota architektury, w większości znana już przez użytkowników.

Opracowali:

Rafał Grabiański

Zbigniew Królikowski