Programowanie kart graficznych

Prof. dr hab. Piotr Białas w trakcie swojej kariery naukowej zajmował się grawitacją symplicjalną, geometrią przypadkową oraz chrodynamiką kwantową na sieci. Obecnie jego zainteresowania przesunęły się bardziej w kierunku informatyki do takich zagadnień, jak metody tworzenia i optymalizacji programów na procesorach graficznych (GPU) i programowanie wielordzeniowych procesorów z instrukcjami wektorowymi. Profesor zapraszany był przez uniwersytety w Amsterdamie, Bielefeld, Barcelonie oraz Saclay. Od wielu lat współpracuje z Wydziałem Fizyki Uniwersytetu Bielefeld.

piotr.bialas@uj.edu.pl

W ostatnich latach widać gwałtowny wzrost zainteresowania zastosowaniem kart graficznych do obliczeń niezwiązanych z grafiką. Wynika to z ogromnej mocy obliczeniowej tych urządzeń, potrzebnej, aby w czasie rzeczywistym *renderować* grafikę na potrzeby coraz bardziej realistycznych gier wideo. Aby temu zadaniu podołać, karty graficzne zostały wyposażone w procesory zawierające ogromne ilości jednostek arytmetycznych. W lepszych kartach ta liczba sięga kilku tysięcy. Daje to tym procesorom graficznym teoretyczną wydajność rzędu kilku teraflopów (flops – floating point operation per second). To od kilkunastu do kilkudziesięciu razy więcej niż potrafią obecne procesory komputerowe (CPU). Należy dodać, że z uwagi na masowość rynku gier, karty graficzne są stosunkowo tanie.

Na drodze do tej obliczeniowej nirwany stoją jednak całkiem potężne przeszkody. Przede wszystkim nie wszystkie problemy da się zrównoleglić. Program wykonujący wiele wątków jednocześnie wymaga zwykle mechanizmów synchronizacji zapewniających, że wątki nie będą sobie przeszkadzać, np. zapisując jednocześnie dane do tego samego miejsca w pamięci. Błędy w takich programach zwykle nie są deterministyczne i są bardzo trudne do znalezienia i naprawienia. Informatycy zajmowali się takimi problemami już od kilkudziesięciu lat, ale dopiero teraz stały się one masowe, ponieważ dotyczą właściwie każdego komputera i każdego oprogramowania.

Wracając do procesorów graficznych, musimy podkreślić ich jeszcze jedną właściwość. Te procesory, z których się składają, nie działają zupełnie niezależnie. Są tak skonstruowane, że pewne ich grupy muszą wykonywać te same czynności, ale z innymi danymi wejściowymi. Nazywane jest to architekturą SIMD (Single Instruction Multiple Data). W kartach NVIDIA najmniejsza taka grupa może liczyć 32 procesory. To rozwiązanie bardzo dobre do przetwarzania grafiki, ale ogranicza rodzaj innych zagadnień, które można prosto zaprogramować na GPU. To jednak tylko wierzchołek

góry zagadnień, które pojawiają się w tzw. High Performance Computing, czyli dziedzinie nauki czy inżynierii, która zajmuje się projektowaniem maszyn i algorytmów do wykonywania obliczeń o dużej złożoności.

Razem z moim doktorantami w Zakładzie Technologii Gier zajmujemy się implementacjami algorytmów na GPU oraz inne architektury, takie jak MIC (Many Integarted Cores) Intela, która zbliżona jest do architektury kart graficznych. Naszym obecnym zadaniem jest opracowanie systemu rekonstrukcji obrazu dla nowego rodzaju Emisyjnego Tomografu Pozytonowego tworzonego w naszym Instytucie przez grupę prof. Pawła Moskala.

Jednym z problemów, z którym się obecnie borykamy, to implementacja algorytmu Monte-Carlo umożliwiającego symulację propagacji kwantów gamma przez detektor. Wydawać by się mogło, że Monte-Carlo to idealne rozwiązanie na procesory równoległe, ponieważ składa się z symulacji ogromnej liczby całkowicie niezależnych zdarzeń. Problem leży w tym, że każda symulowana para kwantów zachowuje się trochę inaczej i może zostać pochłonięta w różnych częściach detektora. W przypadku architektury SIMD wymusza to oczekiwanie całej grupy 32 procesorów, aż zakończy się najdłuższa symulacja. Powoduje to, że program na karcie graficznej wykonuje się jedynie ok. 10 razy szybciej niż na CPU. Właściwa praca, czyli optymalizacja algorytmu, dopiero teraz się zaczyna i mam nadzieję, że niedługo będziemy się już mogli pochwalić lepszymi osiągnięciami. Pozwoli to również przyspieszyć kod na CPU, który również posiada instrukcje SIMD, nazwane AVX, umożliwiające operacje na 8 liczbach zmiennoprzecinkowych naraz. Dlatego to wyzwanie *wektoryzacji* dotyczy właściwie wszystkich grup obliczeniowych i staje się jednym z bardziej palących problemów wielu organizacji.