

Implementacja solwera frontálnego zrównoleglonego w wielowężłowym heterogenicznym środowisku sprzętowym

Paweł J. Wal

Opiekun pracy: dr inż. Łukasz Rauch

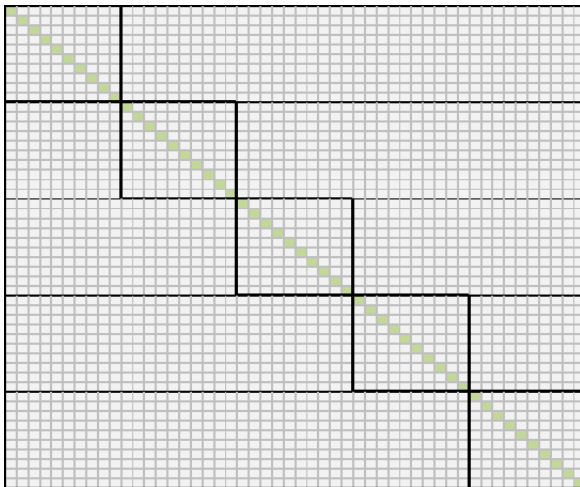
Wydział Inżynierii Metali i Informatyki Przemysłowej
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

7 maja 2015

Oryginalny pomysł - solver multifrontalny na GPU

- Wykorzystany OpenCL
- Masowo równoległe przetwarzanie na GPU
- Wydzielenie frontów rozwiązania
 - Podział macierzy na części (dekompozycja)
 - Wydzielenie z macierzy rzadkiej gęstych fragmentów
- Przetworzenie frontów rozwiązania
 - Przetwarzanie części niezależnie od siebie
 - Rozwiązanie zależności - na koniec

Oryginalny pomysł - dekompozycja



Wady oryginalnego podejścia

- Użycie tylko jednego z dostępnych urządzeń
- Uwarunkowania pamięciowe
 - Ładowanie stosunkowo dużych fragmentów macierzy do pamięci
 - Dodatkowa pamięć potrzebna na specyficzne struktury danych używane przez solwer (mapowanie wierszy)
- Niewielka odporność na zdarzenia katastrofalne (np. awarię zasilania)

Nowy kierunek

- Użycie wszystkich dostępnych urządzeń
- Wykorzystanie możliwości maszyn z wieloma GPU?
 - Wiele GPU = specjalne płyty główne i obudowy = koszt
 - Wiele GPU = specjalne systemy operacyjne i konfiguracje = maszyna o wąskim zastosowaniu
 - Potęga GPU = moc urządzeń klasy konsumenckiej
- Myśl przewodnia: realne zastosowanie w przemyśle i nauce

Założenia projektu

- Możliwość wykorzystania dziesiątek, setek, ... urządzeń dowolnej klasy - przez OpenCL
- Dowolna lokalizacja urządzeń
- Dowolne dopuszczalne obciążenie urządzeń
- Brak nierealistycznych lub wygórowanych założeń (najlepiej: brak założeń w ogóle):
 - Ilość urządzeń w maszynie
 - Moc obliczeniowa urządzeń
 - Rodzaj i jakość połączenia (filtrowane, komórkowe)
 - Czas i wydajność pracy
- Wiele problemów na raz, niezależnie od wielkości
- Odporność na błędy

Inspiracja



Schemat rozwiązania



Przewaga rozwiązań

- Wykorzystanie ukrytego (marnowanego) potencjału urządzeń
- Niewielkie wymagania własne
 - Pracuje z dostępnymi technologiami i popularnymi urządzeniami
 - Nie wymaga inwestycji
 - Nie wymaga specjalistycznej wiedzy by być skutecznie użytkowanym
- Pozwala na rozwiązywanie problemów niepraktycznych lub niemożliwych do rozwiązania

Architektura

- Klient-serwer
- Dlaczego nie MPI?
 - Rozłączne LANy
 - Pojedyncze maszyny w różnych lokalizacjach
 - Gorsze połączenia
- Ostateczna decyzja? Protokół HTTP!
 - Popularność
 - Typowość - raczej nie jest blokowany

Oprogramowanie

- Serwer: Ruby on Rails
 - Jednolite, standardowe API (REST)
 - Bezstanowość
 - Ta sama usługa dostarcza punkt wejścia dla klientów i dla użytkowników (interfejs graficzny)
- Klient: Ruby
 - Dynamiczny rozwój oprogramowania
- Elementy obliczeniowe: C/C++

Przede wszystkim łatwość

- Łatwość rozszerzania serwera - prosty język, jasna struktura danych
- Łatwość modyfikacji klienta
- Łatwość stworzenia zupełnie nowego klienta
- Łatwość dostosowania do aktualnie postawionego celu przez specjalistę
- Łatwość użycia jako gotowego rozwiązania przez specjalistę w innej dziedzinie

Serwer - interfejs użytkownika

Swarmhost - Chromium

Swarmhost

localhost:3000/admin/solutions/new

Aplikacje MadeOnMoon sc2planner mom_admin - Server-generator pegboard nerd » Inne zakładki

SWARMHOST

Logout

Nodes

Solutions

Tasks

New Solution

matrix Nie wybrano pliku

right-hand-side vector Nie wybrano pliku

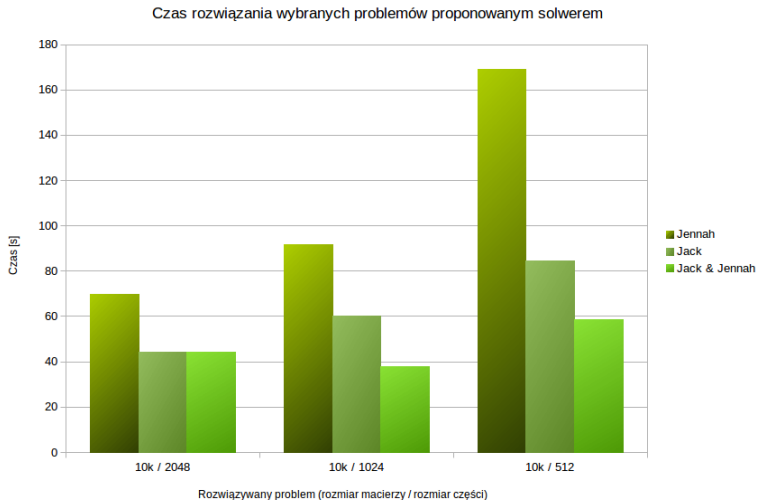
size of part

matrix dimension

Swarmhost © Paweł J. Wal. Insight Template by WebThemez

localhost:3000/admin/nodes

Wyniki dotychczasowych badań



Jack: 4x nVidia Tesla M2090, 1x Intel Xeon X5650. Jennah: 1x nVidia GeForce GT630.
Serwer: 1x Intel Core i7-4710HQ, 16 GB RAM.

Dalszy rozwój

- Rozwiązanie jest nadal w rozwoju
- Aktualnie badane obszary:
 - Usprawnienie komunikacji po HTTP
 - Obniżenie obciążenia serwera
 - Zwiększenie asynchroniczności kluczowych elementów
 - Badanie wpływu parametrów podziału macierzy na prędkość uzyskiwania wyników
 - Badanie wpływu parametrów uruchomieniowych OpenCL na szybkość uzyskiwania rozwiązań na kliencie
 - I inne.