Solwer multifrontalny na platformie GPU

Solwer działa na zasadzie wielu frontów rozwiązania, które są przetwarzane równolegle na karcie graficznej. Oryginalna macierz jest dzielona na bloki. Rozmiar bloku jest dobrany tak, by równał się ilości rdzeni per procesor karty graficznej – jest to optymalizacja mająca na celu maksymalne wykorzystanie dostępnych zasobów urządzenia.

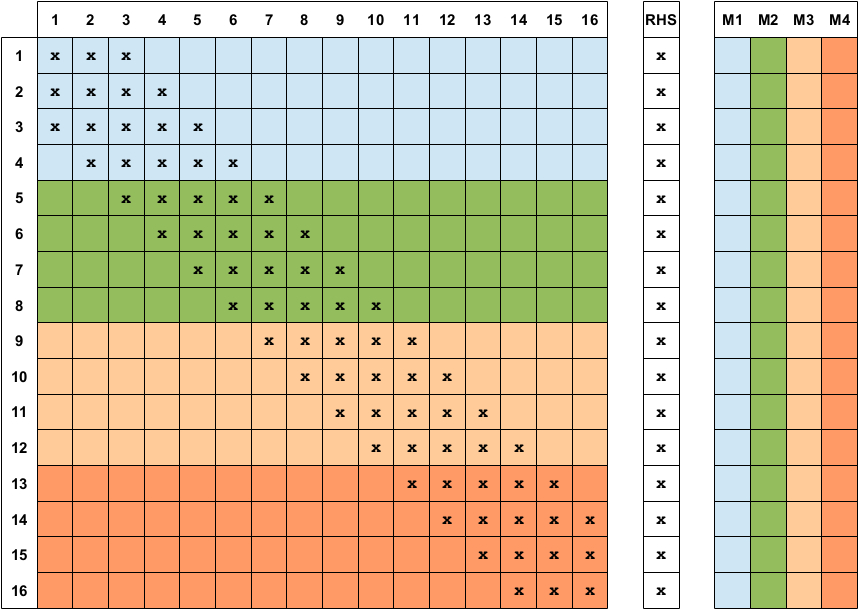
Ze względu na ograniczenia OpenCLa, globalny rozmiar pracy (global work size) musi być wielokrotnością rozmiaru bloku lokalnego (local work size). Przez to ostatni z utworzonych bloków ulega większej dywergencji niż pozostałe (część wątków w ostatnim bloku wychodzi nie wykonując żadnych operacji).

Ogólny algorytm jest to zmodyfikowana wersja eliminacji Gaussa. Celem fazy eliminacji w przód w metodzie Gaussa jest doprowadzenie równania w postaci macierzowej do postaci macierzy schodkowej. W metodzie eliminacji Gaussa jest możliwe zamienianie wierszy miejscami; ze względu na kosztowny (pamięciowo i obliczeniowo) charakter tej operacji, na potrzeby solwera zostały zastosowane symboliczne zamiany wierszy. Przyjęto założenie, iż pierwszy niezerowy wyraz w wierszu macierzy o indeksie n musi występować na pozycji xn. W związku z tym został wprowadzony dodatkowy wektor – swoista mapa rzeczywistej pozycji wiersza w macierzy przechowywanej w pamięci, do pożądanej pozycji wiersza w macierzy schodkowej.

Warto tutaj zauważyć, że w macierzy schodkowej dokładnie jeden wiersz na raz może mieć ten sam pierwszy niezerowy wyraz. O to spostrzeżenie opiera się metoda rozwiązania układu równań.

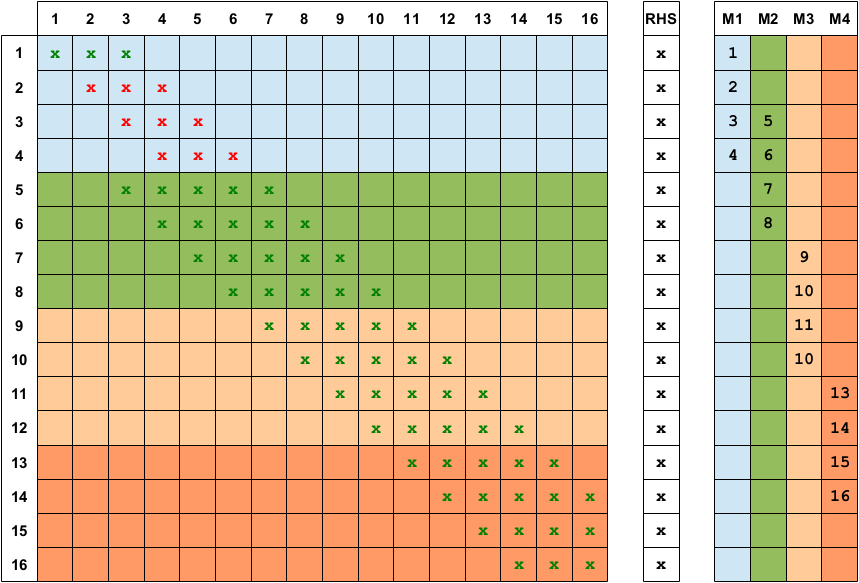
W szerokim ujęciu, rozwiązanie macierzy składa się z co najmniej jednego cyklu rozwiązania. Każdy z takich cykli jest podzielony na dwa etapy. Etapem pierwszym jest przywrócenie spełnienia wyżej wymienionego warunku (każdy wiersz wewnątrz bloku musi mieć unikalny pierwszy wyraz niezerowy). Warto zauważyć, że mimo iż ten warunek zostanie przywrócony wewnątrz bloku, nie ma gwarancji że ten warunek będzie przywrócony pomiędzy blokami; ta sytuacja jest zapewniana dopiero w etapie drugim konkretnego cyklu.

Przyjmijmy, że poszukujemy w zgodzie z tym algorytmem rozwiązania poniższej macierzy. Faktyczne wartości pól zostały zastąpione znakiem x żeby zachować czytelność. Można założyć, że każde pole w którym nie stoi cyfra lub znak x jest równe zeru.



W pamięci urządzenia są tworzone wektory (w ilości równej ilości bloków – tą przykładową macierz podzielono na cztery bloki po cztery wiersze), które będą przechowywały mapowanie. Za pozycję w macierzy uznawana jest pozycja pierwszego niezerowego wyrazu w danym wierszu, natomiast wartość która zostanie umieszczona w takim wierszu to faktyczny indeks wiersza w macierzy.

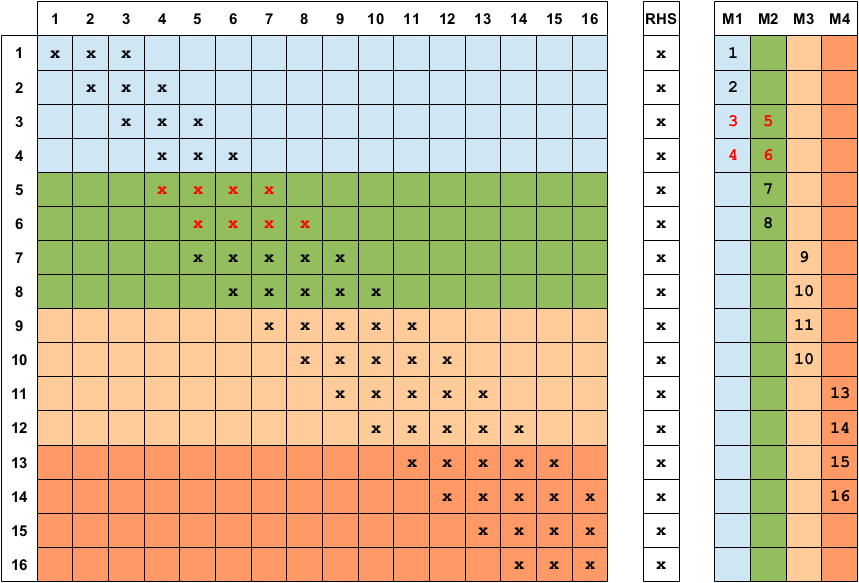
Na tej macierzy przeprowadzamy – równolegle dla czterech bloków – pierwszy etap rozwiązania, tj. przywrócenie warunku unikalności pozycji pierwszego wyrazu niezerowego w danym wierszu w obrębie bloku.



Wiersze oznaczone kolorem czerwonym posiadały nieunikalne w obrębie bloku pierwsze wyrazy niezerowe i zostały sprowadzone (poprzez dodanie do nich wielokrotności innego wiersza, dobranej tak by wyeliminować pierwszy wyraz – w zgodzie z operacjami elementarnymi na macierzach stosowanymi w eliminacji Gaussa) do wartości unikalnych.

Jak wspomniano wcześniej, ta operacja nie daje nam gwarancji że warunek unikalności będzie spełniony na przestrzeni całej macierzy. W istocie, rozpatrując wektory map poziomo – wierszami – łatwo zauważyć, że identyczny indeks pierwszego wyrazu niezerowego mają wiersze 3 i 5, oraz wiersze 4 i 6.

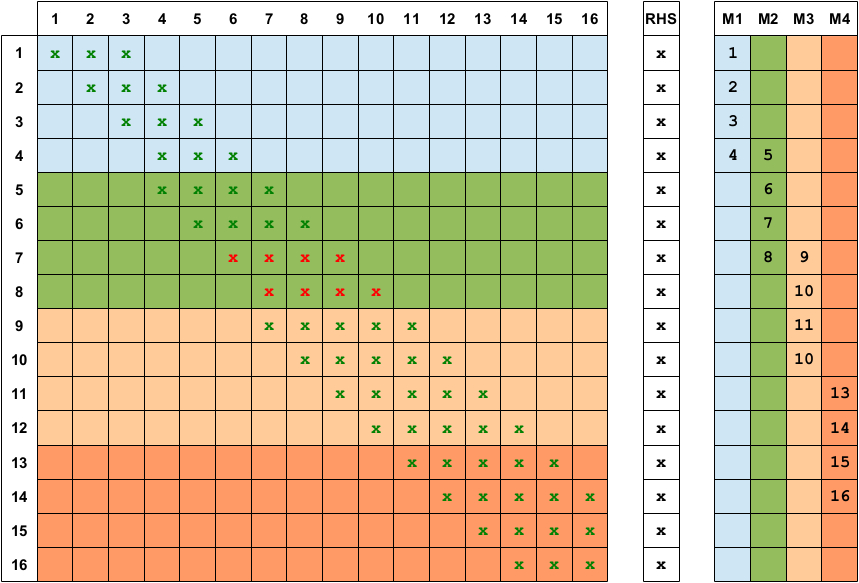
Za rozwiązanie tego problemu odpowiada drugi etap rozwiązania (zrealizowany jako drugi kernel OpenCL), który dokonuje właśnie operacji sprawdzenia, czy dla każdego wiersza w mapie istnieje jedna lub mniej wartość. Jeśli istnieje więcej niż jedna, na nadmiarowych wierszach jest wykonywana analogiczna procedura eliminacji jak opisano powyżej.



Kolorem czerwonym w macierzy zostały oznaczone wiersze które zostały zredukowane. Kolorem czerwonym w wektorach zostały oznaczone numery wierszy które zostały uznane za kinfliktujące i została na nich przeprowadzona redukcja.

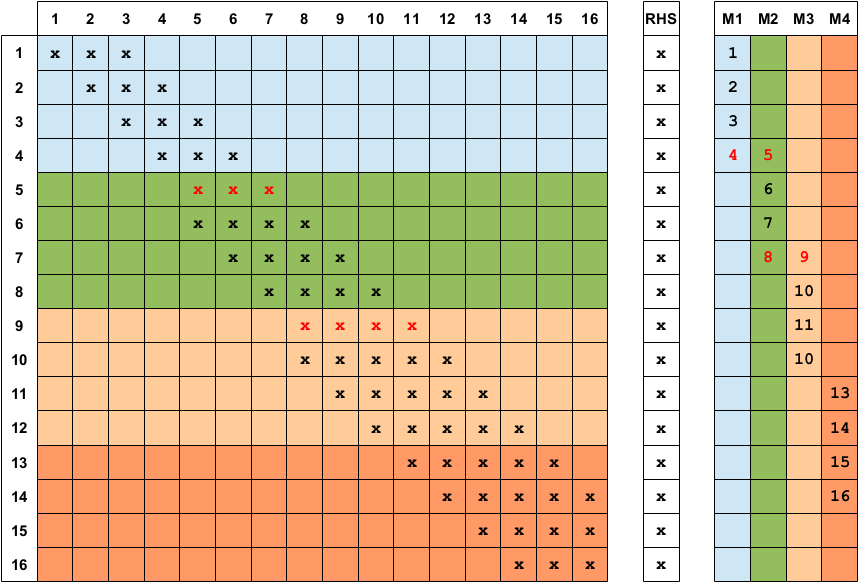
Warunkiem zakończenia rozwiązania i uznania wyjścia z solwera za rozwiązanie macierzy jest brak operacji eliminacji wykonanych w drugim etapie cyklu rozwiązania. W tym przypadku wykonano dwie operacje, co oznacza że konieczne jest wykonanie kolejnego cyklu i ponowne przeprowadzenie obydwu etapów.

Jak można zauważyć na rysunku, zredukowanie konfliktów między mapami (określanych dalej jako konflikty globalne) wprowadziło nowe konflikty wewnątrz bloków (określane dalej jako konflikty lokalne). Pierwszy etap następnego cyklu mógłby przebiec następująco:



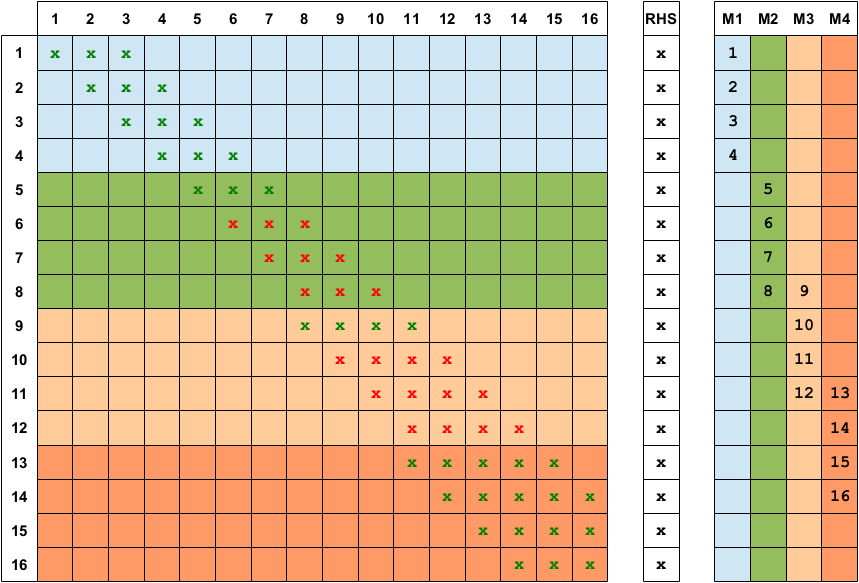
Bloki 1, 3 i 4 szybko zakończą swoje działanie, gdyż nie znajdą konfliktów lokalnych. W bloku numer 2, oznaczonym kolorem zielonym, na czerwono zaznaczono zawartość wierszy które muszą ulec eliminacji by rozwiązać konflikt lokalny.

Rozwiązanie tego konfliktu lokalnego wprowadziło dwa nowe konflikty globalne, które zostaną rozwiązane w drugim etapie tego cyklu rozwiązania.

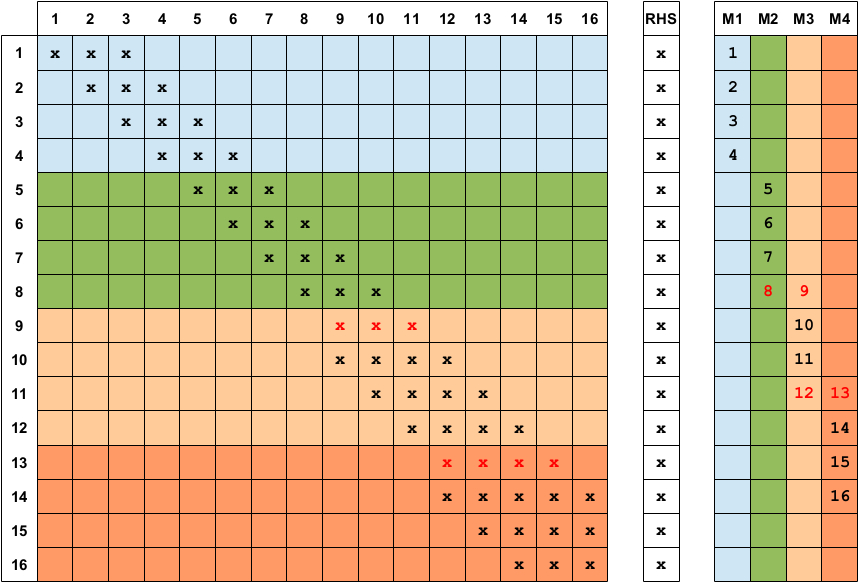


Kolorem czerwonym zostały oznaczone wiersze w których wystąpiły konflikty globalne, które zostały rozwiązane w drugim etapie tego cyklu.

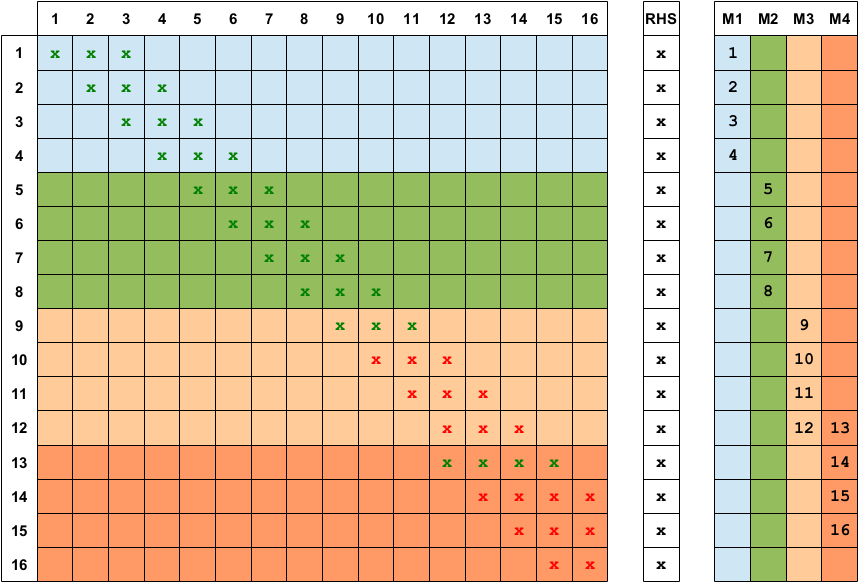
Ponieważ nadal liczba operacji wykonanych przez etap drugi tego cyklu jest większa niż zero, konieczne jest przeprowadzenie następnego cyklu.

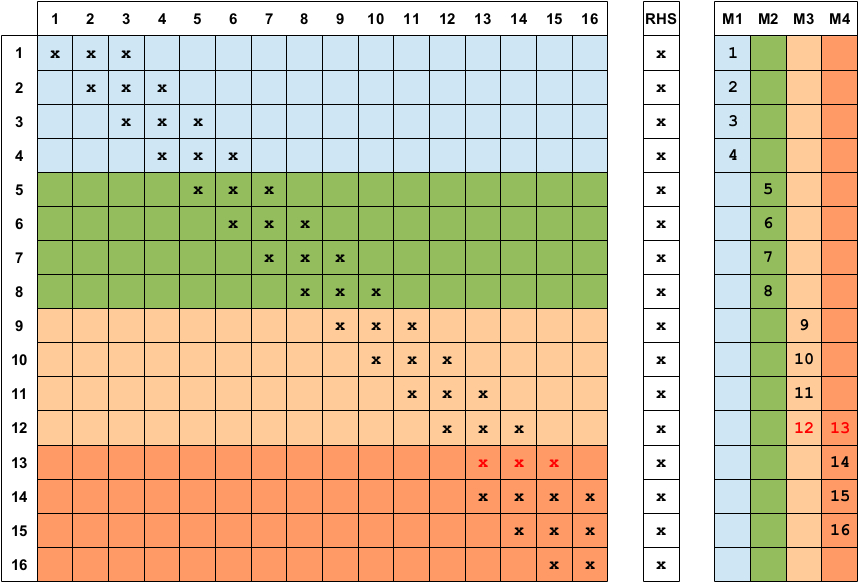


Kolejna faza rozwiązywania konfliktów lokalnych wprowadza nowe konflikty globalne:

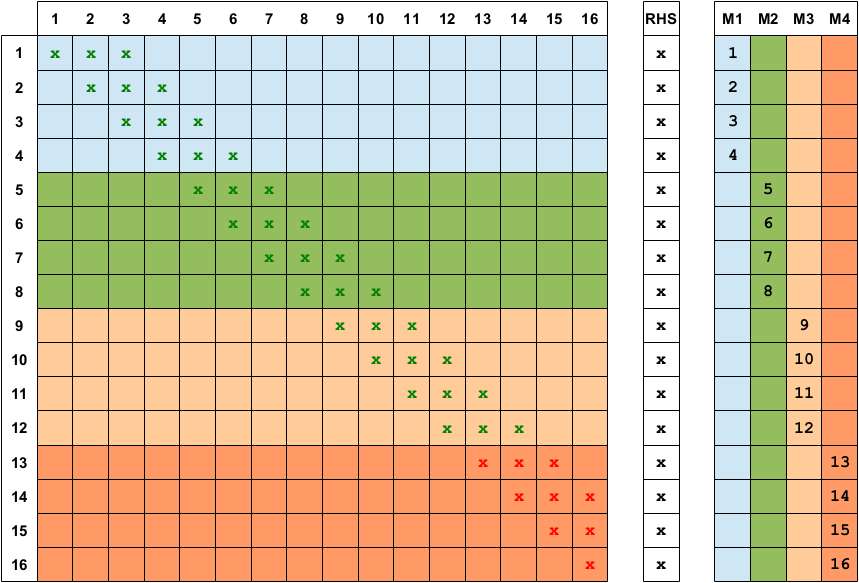


Ponieważ nadal nie uzyskaliśmy zerowej ilości operacji w drugim etapie, przeprowadzany jest kolejny cykl rozwiązania:





Po tej akcji, w następnym cyklu, redukcja konfliktów lokalnych nie daje już nowych konfliktów globalnych:



Etap drugi – rozwiązywanie konfliktów globalnych – nie wykonuje już żadnych operacji, co oznacza że można zakończyć pętlę cykli rozwiązania, pobrać z urządzenia dane i wykonać – sekwencyjnie już, na CPU – fazę podstawienia wstecz eliminacji Gaussa, co w konsekwencji skutkuje rozwiązaniem.

Warto zauważyć, że na powyższych rysunkach – dla spójności i jasności wyjaśnień – zostało przedstawione rozwiązanie przeprowadzone użytym algorytmem, ale tak, jak wykonałoby je urządzenie sekwencyjne (zawsze wiersz, który jest niżej zostanie zredukowany). W przypadku urządzeń GPU nie ma gwarancji który wątek zostanie wykonany pierwszy, w związku z czym faktyczna postać rozwiązania na GPU ma znacznie mniej określoną postać (dobór kolejności wątków do wykonania jest procesem probabilistycznym).