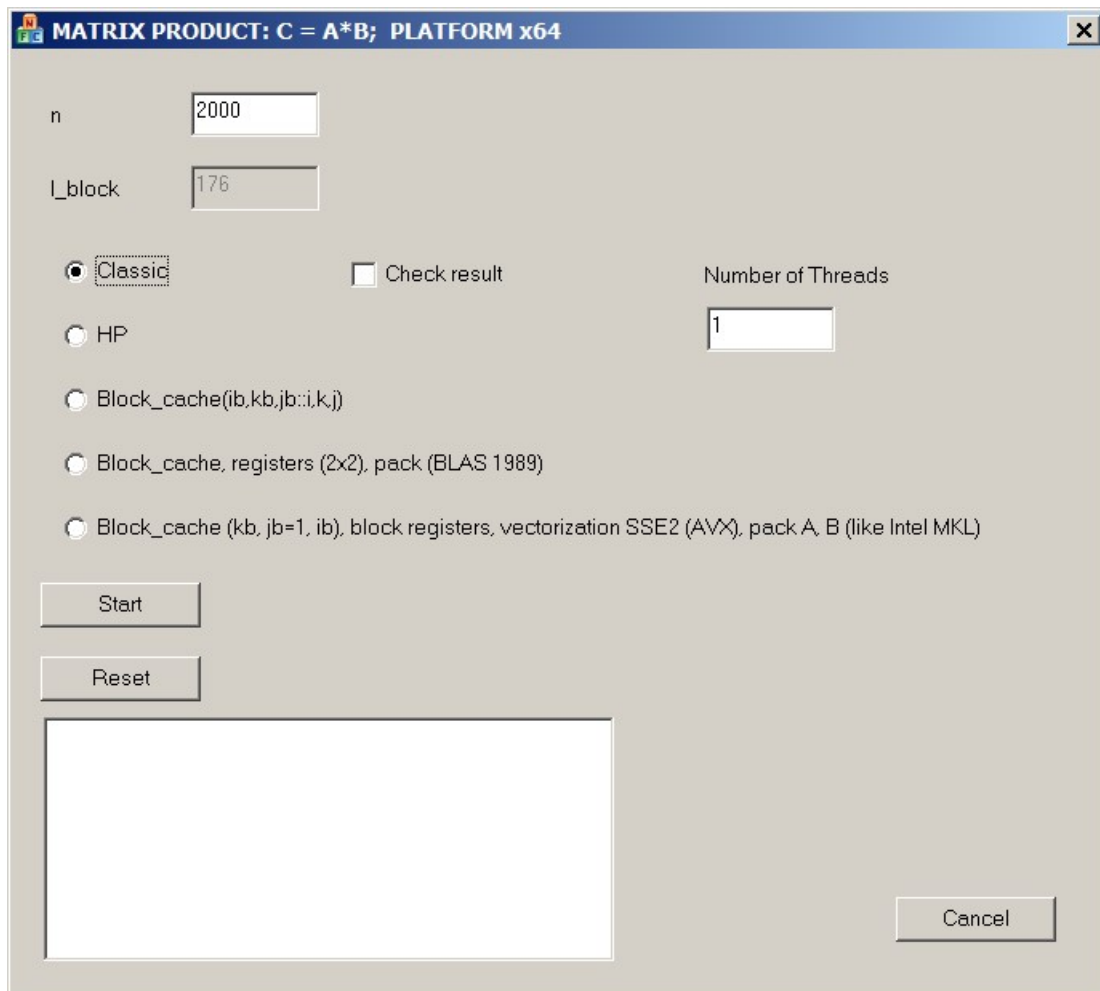


LAB_3_1

$$C = A * B$$

Porównywanie różnych metod mnożenia macierzy przez macierz

1. Zadanie laboratoryjne realizuje test $A \cdot B = C$, gdzie A, B, C – macierzy kwadratowe o rozmiarze $n \times n$. Nazwy metod oraz zrealizowane algorytmy ściśle odpowiadają materiałom wykładu W3 oraz przedstawionemu na stronie www podręczniku elektronicznemu. Macierzy A, B, C są umieszczone w pamięci komputera wiersz po wiersze.
2. Stworzyć katalog o dowolnej nazwie i rozpakować archiwum pobrany ze strony www.
3. Uruchomić Mulm_.exe:



Zrealizowane metody:

Metoda	Krótki opis	Wielowątkowość
Classic	Metoda klasyczna (i, j, k)	NIE
HP	Modyfikacja metody klasycznej (i, k, j) – usunięcie skoków przy pobieraniu elementów macierzy B	TAK
Block cache $(ib, kb, jb::i, k, j)$	Podział każdej z macierzy na bloki o rozmiarze $l_block \times l_block$. Kolejność indeksów dla bloków jest ib, jb, kb ; indeksów w bloku: i, j, k .	TAK
Block cache, register (2×2) , pack (BLAS 1989)	Zastosowane blokowanie na poziomie RAM – cache – (podział każdej z macierzy na bloki o rozmiarze $l_block \times l_block$), blokowanie rejestrów 2×2 , pakowanie bloku macierzy B tak, żeby uniknąć skoków w danych przy pobieraniu elementów tej macierzy. Kolejność indeksów: $kb, ib, jb::j, i, k$	NIE
Block cache $(kb, jb=1, ib)$, block YMM (XMM) registers $(2 \times 4 \times 8)$, SSE2 (AVX, FMA), pack A, B (similar to Intel MKL library)	Blokowanie cache ($l_block = 176$), jest używane blokowanie rejestrów wektorowych. W technice SSE2 schemat blokowania rejestrów jest $4 \times 4 \times 8$ (4 słowa double A , 4 słowa double – B , 8 razy rozwijana pętla wewnętrzna po indeksu k). Używane instrukcje SSE2 (Streaming SIMD Extension). Bloki macierzy A, B są przepakowane tak, żeby: a) dane, załadowane w cache, mają pozostawać tam jak najdłużej (minimalizacja przeładowania cache) b) minimalizacja read miss W technice AVX (AVX+FMA) schemat blokowania rejestrów jest $8 \times 16 \times 4$ (8 słów double A , 16 słów double – B , 4 razy rozwijana pętla wewnętrzna po indeksu k). Schemat blokowania cache i rejestrów jest podobny do używanego w Intel MKL library.	TAK

4. **Check result** – testuje wyniki na podstawie porównywania z wynikami metody Classic. **Uwaga! To powoduje istotne zwiększenie czasu obliczeń, jest rekomendowane dla $n \sim 500$.**
5. **Number of Threads** – liczba wątków dla metod, które wspierają wielowątkowość.
6. **Start** – uruchomić obliczenia

7. **Reset** – po zakończeniu obliczeń przestawić program do kolejnego obliczenia. Wyniki obliczeń poprzednich znikają.

8. **Zadanie.** *Wszystkie testy wykonujemy na jednym wątku. Każdy test wykonuje się co najmniej 3 razy.* Wprowadzony rozmiar zadania po zakończeniu obliczeń będzie zmieniony tak, że $n \% 8 = 0$. Rejestrujemy wydajność, opierając na ten fakt, że **przy wystarczająco dużym rozmiarze zadania wydajność nie zależy od rozmiaru zadania n . Nadaje nam to prawo porównywać wydajności (tylko nie czasy obliczeń!) różnych metod przy różnych rozmiarach zadań n pod warunkiem, że n jest wystarczająco duże.**

Dolna granica „wystarczająco dużego” n zależy od metody i od wydajności procesora. Dla każdej metody będzie podany rozmiar zadania n dla testowania.

8.1. Metoda klasyczną. Rozmiar zadania $n = 1000$.

8.2. Metoda HP. Rozmiar zadania $n = 1000$.

8.3. Metoda blokowa – block cache. Rozmiar zadania $n = 1000$.

Tu trzeba na podstawie numerycznego eksperymentu dobrać optymalny rozmiar bloku l_{block} . Sprawa w tym, że wzór teoretyczny $l_b = \sqrt{(M/3)}$ (patrz wykład W3) jest wyprowadzony z założenia że pamięć cache jest jednorodna i ma tylko jeden poziom. Jednak współczesne procesory mają trzy poziomy pamięci cache, przy czym prędkość dostępu do danych różnych poziomów cache znacznie się różni.

Dlatego tworzymy wykres $W = W(l_b)$, gdzie W – wydajność (przypomnieć definicje wydajności!), l_b – rozmiar bloku. Zaczynamy od klasycznego wzoru $l_b = \sqrt{(M/3)}$, korzystając programem CPU-Z dla wyznaczenia wartości M przy założeniu, że M jest pojemnością cache dla danych poziomu L1¹. Tworzymy pierwszy punkt naszego wykresu. Dalej wyznaczamy wydajność dla $l_b = 20, 40, 80, 100, 120, 140, 160, 180, 200$. W podanych granicach wybieramy to znaczenie l_b , które doprowadzi do większej wydajności.

8.4. Block cache, register (2×2), pack (BLAS 1989). Rozmiar zadania $n = 1000$.

Powtarzamy badania, opisane dla poprzedniej metody.

8.5. Metoda wykonana w technice Intel MKL. Rozmiar zadania $n = 4000$.

Ta metoda wykorzystuje mikrojądro

(<https://repozytorium.biblos.pk.edu.pl/resources/33553>), które było stworzone przy tworzeniu solwera wielofrontalnego programu MES (metoda elementów skończonych) o nazwie SCAD (www.scadsoft.com). Później, kiedy pojawili się procesory wspierające technikę AVX, oznaczone mikrojądro uległo zmianie (<https://annals-csis.org/proceedings/2013/pliks/98.pdf>). Przyczyna, która wymusiła autora stworzyć wymienione powyżej mikrojądro, polega na tym, że procedury *dgemv* (general matrix multiplication) zrealizowane w bibliotekach wysokiej wydajności Intel MKL, AMD ACML i w innych, nawet dla macierzy symetrycznych wymagają alokowania pamięci o rozmiarze n^2 , chociaż macierz symetryczna potrzebuje przechowywać $n \cdot (n+1)/2 \approx n^2/2$ dla dużych n .

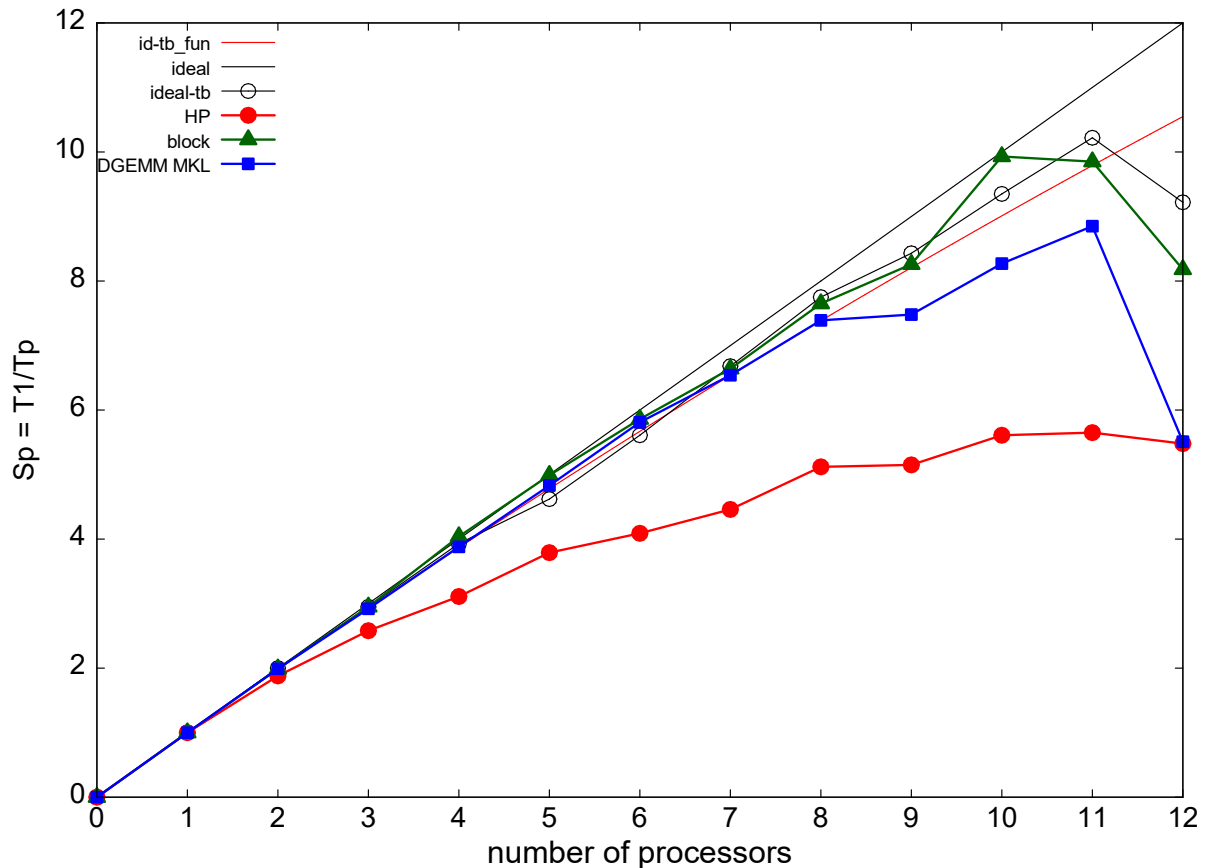
Żeby wyjaśnić optymalny rozmiar bloku, powtarzamy badania, wykonane dla poprzednich metod blokowych.

9. Przedstawić sprawozdanie, zawierające wyniki testów oraz wykresy $W = W(l_b)$.

¹ Proszę wyznaczyć M w słowach double, a nie w bajtach! Proszę nie mnożyć wynik CPU-Z na liczbę rdzeni fizycznych procesora, ponieważ cache L1 jednego procesora nie ma prawa wtrącić się do cache L1 drugiego procesora!

Dodatek

Wyniki testów w trybie wielowątkowym na komputerze klasy workstation z dwoma 6-rdzeniowymi procesorami **Intel Xeon X5660 @ 2.8 GHz / 3.2 GHz**, **RAM 24 GB, DDR3**, **platforma x64**.



Te procesory używają tryb turbo-boost: przy obciążeniu niewielkiej ilości rdzeni częstotliwość procesora jest podniesiona do 3.2 GHz; przy obciążeniu dużej ilości rdzeni częstotliwość procesorów spada do wartości nominalnej – 2.8 GHz. Dla tego prosta "ideal" nie opisuje idealnego przyspieszenia systemu równoległego na takim komputerze, ponieważ nie bierze pod uwagę zmienności częstotliwości procesorów.

Krzywa "id-tb fun" jest aproksymacją idealnego przyspieszenia w trybie turbo boost – jest to parabola kwadratowa, która przechodzi przez punkty (0; 0), (1;1), (12; 10.5), gdzie wartość $10.5 = 12 \cdot 2.8 / 3.2$ (zakładamy że przy obciążeniu wszystkich rdzeni częstotliwość procesora zmniejszy się do 2.8 GHz, a przy obciążeniu tylko jednego rdzenia częstotliwość wynosi 3.2 GHz).

Krzywa "ideal-tb" jest wynikiem testowania programu, który intensywnie liczy funkcje trygonometryczne i praktycznie nie obciąża układ pamięci. Można uważać, że speed-up dla takiego algorytmu będzie bardzo przybliżony do przyspieszenia idealnego w trybie turbo boost.

Krzywy "HP", "block", "DGEMM MKL" odpowiadają metodom HP, block cache i procedurze DGEMM z biblioteki Intel Math Kernel Library.

Wyraźnie widać, że metoda naiwna HP znacznie ustępuje w skalowalności metodom blokowym.

A to – maksymalna wydajność, osiągnięta dla każdej metody na tym komputerze:

Metoda	Największa wydajność, MFLOPS
HP	8 817
Block cache	17 496
DGEMM MKL	104 879

