Maciej Matyka
Wydział Fizyki i Astronomii
Instytut Fizyki Teoretycznej
Uniwersytet Wrocławski
oraz Vratis Sp. z o. o.

Zbigniew Koza
Wydział Fizyki i Astronomii
Instytut Fizyki Teoretycznej
Uniwersytet Wrocławski
oraz Vratis Sp. z o. o.

Łukasz Mirosław
Instytut Informatyki
Politechnika Wrocławska
oraz Vratis Sp. z o. o.

Wydajność otwartych implementacji metody sieciowej Boltzmanna na CPU i GPU

STRESZCZENIE

Metoda sieciowa Boltzmanna (LBM) służy do symulacji dynamiki płynów. Celem pracy jest zbadanie wydajności jej dwóch otwartych implementacji równoległych: Palabos i Sailfish. Pierwsza uruchamia się na klasycznym procesorze wielordzeniowym (multi-CPU), druga na pojedynczym układzie graficznym (GPU). Dyskutujemy, w jaki sposób na ich wydajność wpływają takie czynniki, jak zastosowana precyzja obliczeń, rodzaj zagadnienia, warunki fizyczne oraz architektura procesora. W każdym zbadanym przypadku kod uruchomiony na karcie graficznej jest szybszy od kodu uruchomionego na wielordzeniowym CPU. Uzyskane przyspieszenia wynoszą od 3x do 12x dla podwójnej i od 24x do 42x dla pojedynczej precyzji obliczeń.

WSTĘP

Symulacje zjawisk fizycznych należą do jednych z najciekawszych i zarazem najbardziej wymagających obliczeniowo zastosowań współczesnych komputerów. Dzięki szybkiemu postępowi zarówno w dziedzinie algorytmiki, jaki i projektowania sprzętu, metody symulacji są obecnie jedną z najdynamiczniej rozwijających się dziedzin nauki, a zakres ich zastosowań jest już tak szeroki, iż w praktyce osiągnęły one status trzeciego filaru nauki – obok eksperymentu i teorii.

Jedną z najważniejszych gałęzi metod symulacji jest obliczeniowa dynamika płynów (ang. *Computational Fluid Dynamics* – CFD) mająca liczne zastosowania w takich dziedzinach, jak inżynieria lotnicza i kosmiczna, przewidywanie pogody, ocena zabezpieczenia przeciwpożarowego budynków, sport czy medycyna. W ostatnich latach pojawiły się liczne prace wykorzystujące CFD w diagnostyce różnego rodzaju chorób w kardiologii, kardiochirurgii czy też pulmonologii.

Przykładem zastosowania CFD w kardiologii jest symulacja przepływu krwi przez lewą komorę serca, którą można znaleźć u Longa [1, 2], gdzie wykorzystano skany MRI do rekonstrukcji geometrii komory. W tego typu zagadnieniach często trzeba uwzględnić zarówno elastyczność naczynia krwionośnego, które ulega deformacji pod wpływem przepływu krwi, jak i sam przepływ. Dlatego też modele hydrodynamiki np. serca muszą jednocześnie wyznaczać zarówno przepływ krwi, jak i odkształcenia samego naczynia [3]. Więcej informacji na temat modelowania przepływu krwi przez serce można znaleźć np. w [4] . Przykładem zagadnienia medycznego, w którym symulacje mogą mieć w przyszłości znaczenie krytyczne, są badania nad przepływem krwi w naczyniach krwionośnych narażonych na ryzyko wystąpienia tętniaków, np. w aorcie brzusznej [5, 6]. W tych przypadkach dąży się do opracowania wiarygodnych procedur szacowania prawdopodobieństwa, z jakim może dojść do pęknięcia tętniaka tego naczynia. Z kolei w dziedzinie pulmonologii pojawiły się prace dotyczące symulacji przepływu powietrza przez dolne drogi

oddechowe [7] bazujące na obrazach uzyskanych metodą tomografii komputerowej. W ramach tych prac zbadano m.in. wpływ przepływu krwi na deformację ściany naczynia (ang. *Fluid-Structure Interaction*), jak również możliwość deformacji geometrii w czasie oddychania. Modele takie mają zastosowanie głównie w diagnostyce pacjentów podłączonych do respiratorów jak i przy poważnych schorzeniach płuc. Analiza przepływu powietrza w płucach i związanego z tym przepływu leków w postaci aerozolu ma też duże znaczenie dla przemysłu farmaceutycznego [7].

Symulacje CFD wykorzystuje się też do symulacji procesu angiogenezy, czyli procesu tworzenia się nowych naczyń pod wpływem progresji tkanki nowotworowej. Badania takie mogą być bardzo pomocne jako dodatkowe narzędzie ułatwiające zarówno zaplanowanie odpowiedniej terapii, jak i dokonanie oceny potencjalnego jej wpływu na stan pacjenta. W ostatnich latach pojawiła się też seria artykułów na temat dynamiki chemioterapii [8] i symulacji przebiegu radioterapii [9, 10]. Modelowany był m.in. wpływ kinetyki farmakologicznej na rozwój nowotworu [11].

Zastosowania medyczne stanowią szczególne wyzwanie dla metod obliczeniowych, gdyż wymagają znalezienia równowagi między dokładnością rozwiązania, jego praktyczną użytecznością, nakładem pracy ze strony personelu medycznego i czasem potrzebnym do uzyskania wyników [5, 12]. Tylko symulacje o dostatecznej dokładności dające wyniki weryfikowalne doświadczalnie i otrzymywane w czasie akceptowalnym z medycznego punktu widzenia mogą być obiektem zainteresowania lekarzy. W praktyce klinicznej zapewne preferowane będą obliczenia wykonywane na komputerach osobistych – rozwiązanie na pewno wygodniejsze i tańsze od symulacji uruchamianych na klastrze czy superkomputerze. Osiągnięcie tego celu wymaga jednak zastosowania sprzętu o ogromnej mocy obliczeniowej i nowoczesnych, szybkich algorytmów.

Obiecującą platformą sprzętową, która może podołać wymaganiom stawianym wielu symulacjom medycznym, są współczesne karty graficzne (*Graphics Processing Unit*, GPU). Dzięki masowo równoległej architekturze, układy te charakteryzują się znacznie większą teoretyczną mocą obliczeniową oraz wyraźnie większą przepustowością zintegrowanej pamięci operacyjnej w porównaniu do nawet najszybszych klasycznych procesorów wielordzeniowych (CPU). Z kolei w dziedzinie algorytmiki jedną z najbardziej obiecujących metod symulacji transportu płynów w układach o złożonej geometrii jest metoda gazu sieciowego Boltzmanna (ang. *Lattice Boltzmann Method*, LBM). Dzięki względnej prostocie implementacji skomplikowanych warunków brzegowych oraz ich lokalności metoda ta dość dobrze skaluje się w implementacjach współbieżnych. Metoda LBM znalazła już zastosowanie w symulacjach przepływów w układach o znaczeniu biologicznym i medycznym (np. [13, 14]).

Głównym celem niniejszego artykułu jest analiza porównawcza wydajności dwóch pakietów otwartego oprogramowania do symulacji przepływów metodą LBM: Palabos i Sailfish. Pierwszy z nich przeznaczony jest do uruchamiania na klasycznym procesorze wielordzeniowym (lub klastrze utworzonym z takich komputerów), druga – na karcie graficznej. W szczególności chcemy odpowiedzieć na pytanie, czy przy założeniu, że symulacja uruchamiana jest na pojedynczym komputerze osobistym, implementacja wykonywana na GPU może być istotnie szybsza od implementacji uruchomionej na wielordzeniowym CPU.

METODA SIECIOWA BOLTZMANNA

Metoda sieciowa Boltzmanna (LBM) jest jedną z najnowszych metod komputerowych używanych do rozwiązywania zagadnienia transportu płynów. Historycznie metoda ta wywodzi się z automatu komórkowego FHP [15, 16], w którym płyn przybliża się cząstkami, których wektory prędkości w dowolnej chwili mogą przyjmować jedną z zaledwie siedmiu prędkości. W metodzie LBM zmienne dyskretne określające prędkości cząstek zostały zastąpione siedmioma zmiennymi ciągłymi interpretowanymi jako prawdopodobieństwa, że cząstki podążają w określonym kierunku na sieci. Znając te funkcje, można wyznaczyć parametry makroskopowe przepływu takie jak

lokalna prędkość i gęstość płynu. Istotą metody Boltzmanna jest prosty algorytm (typu automatu komórkowego), w którym ewolucję funkcji rozkładu wykonuje się w dwóch krokach: przesunięcia i kolizji. Człon przesunięcia jest trywialny (cząsteczki płynu „przesuwa się" do sąsiednich węzłów siatki obliczeniowej), a człon kolizji w podstawowej wersji (tzw. metoda BGK) przybliża się, zakładając liniową relaksację funkcji rozkładu prawdopodobieństwa do wartości równowagowych. Funkcję równowagową wyznacza się metodami fizyki statystycznej z rozkładu Maxwella-Boltzmanna.

W celu polepszenia własności numerycznych modelu LBM powstało wiele jego wariantów, np. model nieściśliwy [17], model zachowawczy dla mikroprzepływów [18] czy model entropowy [19]. Dodatkowo opracowane zostały wielo-relaksacyjne modele kolizji (ang. *Multi Relaxation Time* – MRT) [20], które dodatkowo zwiększają dokładność i stabilność metody. Podstawowe wiadomości o modelach LBM, ich implementacjach, zastosowaniach w problemach technicznych i naukowych oraz przegląd najnowszych osiągnięć znaleźć można w bogatej literaturze (np. [21, 22, 23, 24]).

PALABOS

Istnieje kilka otwartych implementacji metody LBM w wersji na procesory klasyczne. Najbardziej znanymi są OpenLB [25] i Palabos [26]. Obie biblioteki napisano obiektowo w języku C++, obie mają porównywalne możliwości. Do testów wydajności wybrany został Palabos. Jest to otwarta, darmowa (dostarczana na licencji GPLv3) implementacja wysokiego poziomu (kod C++ intensywnie wykorzystujący szablony). Palabos posiada zaimplementowane podstawowe siatki obliczeniowe LBM (przepływy dwuwymiarowe: d2q9; trójwymiarowe: d3q13/15/19/27), w tym sieci z różnym stopniem podziału, modele przepływów (termalny, nienewtonowski) oraz operatory kolizji (BGK, MRT).

W Palabos obliczenia równoległe realizowane są za pomocą technologii MPI (ang. *Message Passing Interface*), co pozwala na uruchamianie symulacji na klastrach obliczeniowych z pamięcią rozproszoną. Obliczenia można też prowadzić w mniejszej skali na komputerze osobistym (architektura wielordzeniowa z pamięcią współdzieloną). Testy poprawności i efektywności kodu tej biblioteki zostały opublikowane w artykułach dotyczących m.in. wyznaczania przepływu krwi przez naczynia krwionośne [14] czy przepuszczalności w ośrodkach porowatych [27].

SAILFISH

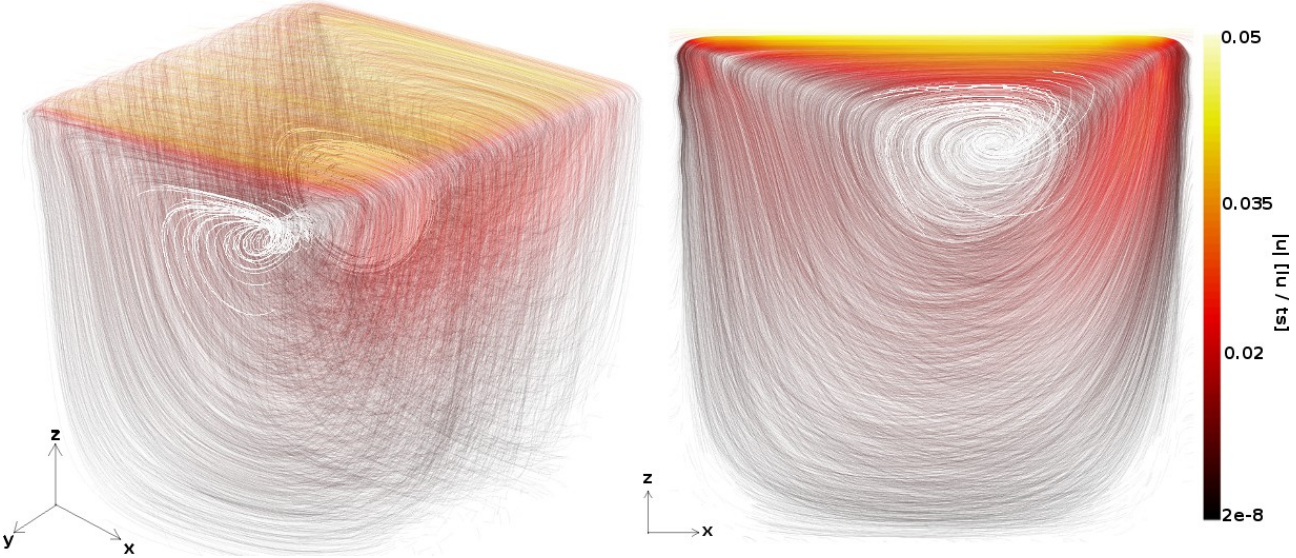
Biblioteka Sailfish [28] jest to implementacja metody LBM dla kart graficznych (GPU) opracowana w Instytucie Fizyki Uniwersytetu Śląskiego w Katowicach. Napisano ją z użyciem kilku języków i technik programistycznych (Python, C++, szablony Mako), dzięki czemu możliwe jest generowanie w locie programów dla GPU w technologiach CUDA lub OpenCL. Podobnie jak Palabos, Sailfish zawiera implementację podstawowych siatek obliczeniowych metody LBM oraz kilka modeli przepływów i operatorów kolizji (BGK, MRT), jednak nie zaimplementowano w nim jeszcze możliwości zagęszczenia sieci. Mimo że Sailfish jest projektem bardzo młodym, posiadającym wciąż status kodu eksperymentalnego, jest on już znany w środowisku zajmującym się rozwojem i zastosowaniami modelu LBM [29, 30].

W dalszej części artykułu wszędzie, gdzie będzie mowa o CPU, będziemy mieli na myśli kod Palabos uruchomiony na pojedynczym procesorze wielordzeniowym. Wszędzie, gdzie będzie mowa o GPU, będziemy mieli na myśli kod biblioteki Sailfish pracujący w środowisku nVidia CUDA na jednej karcie graficznej.

SPRZĘT

Do badania wydajności kodu CPU wykorzystaliśmy węzeł Sun Fire X2200 z procesorem Quad-Core AMD Opteron 2384 taktowanym zegarem 2.7 Ghz i wyposażonym w 15 GB pamięci.

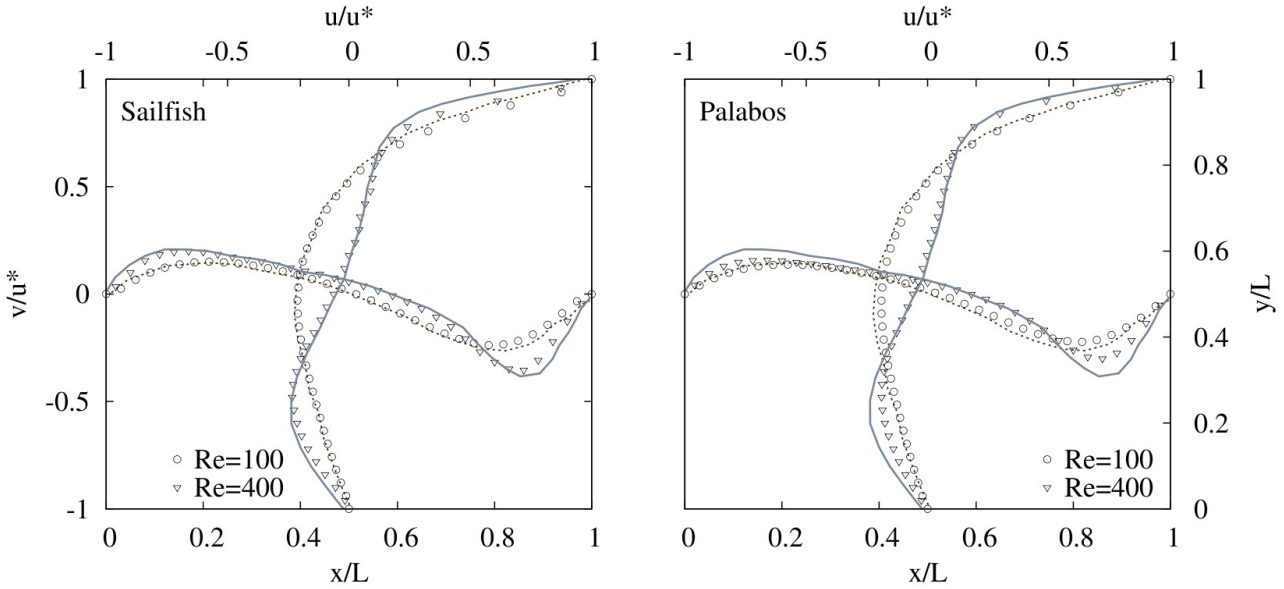
Kod GPU testowaliśmy na na dwóch kartach: a) GTX 460 (2 GB pamięci DDR5, 115.2 GB/s; 336 rdzeni CUDA taktowanych zegarem 675 MHz); b) Tesla C 2070 (6 GB pamięci DDR5, 144 GB/s, ECC off; 448 rdzeni CUDA, zegar 1.15 GHz). Teoretyczna moc obliczeniowa użytego procesora Tesla wynosi 1.03 Teraflop/s w pojedynczej i 515 Gigaflop/s w podwójnej precyzji.



Ryc. 1. Wizualizacja przepływu przez trójwymiarową komorę wyznaczonego kodem Sailfish dla liczby Reynoldsa Re=100. Jasność koloru odpowiada wartości prędkości (im jaśniejszy, tym większa prędkość). Na biało zaznaczyliśmy linie prądu w okolicy wiru centralnego. (rycina kolorowa).

WYNIKI

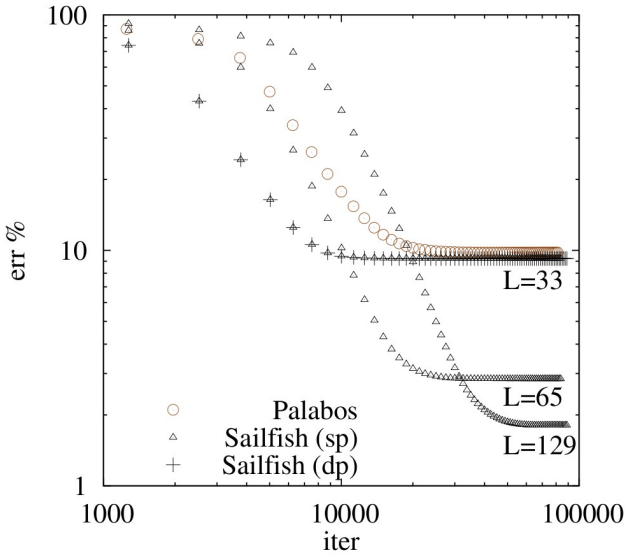
Do testowania wydajności implementacji metody LBM wybraliśmy przepływ przez trójwymiarową, prostopadłościenną komorę (ang. *cavity*) o szerokości L, w której pięć ścian jest nieruchomych, natomiast górna ściana (pokrywa) porusza się ruchem jednostajnym w kierunku osi x. W tych warunkach na brzegach nieruchomych ścian płyn jest nieruchomy, natomiast na brzegu pokrywy posiada stałą, niezerową prędkość skierowaną wzdłuż pokrywy (sytuacja analogiczna jak np. w [31]). Na ryc. 1 przedstawiamy wizualizację linii prądu dla danych otrzymanych przy pomocy kodu GPU. Na ryc. 2 przedstawiamy porównanie wyników przepływu przez tę komorę



Ryc. 2. Przepływ przez trójwymiarową komorę wykonany kodem Sailfish (lewy) i Palabos (prawy) dla L=33. Wyniki porównane do [31] dla Re=100 (linia przerywana) i Re=400 (linia ciągła).

otrzymanych metodą LBM na GPU oraz na CPU ($3 \cdot 10^4$ iteracji). Wyniki metody LBM porównujemy ze standardową metodą spektralną opartą na wielomianach Czebyszewa [31, 32]. W obu przypadkach (CPU i GPU) rozdzielczość siatki obliczeniowej była porównywalna (około 32^3 komórek). Widać wyraźnie, że zgodność wyników obu kodów LBM z wynikiem wzorcowym jest najlepsza w środku układu. Lepsze dopasowania uzyskuje się dla też dla niższych liczb Reynoldsa.

Aby sprawdzić, jak zachowuje się metoda LBM dla różnej liczby iteracji wyrysowaliśmy różnice między składową x prędkości up w punkcie (0, 0.853) i wyznaczoną w [31] wartością wzorcową $u^* = -0.383$ (ryc. 3) dla różnej liczby iteracji i różnych rozmiarów siatki obliczeniowej.



Ryc. 3. Porównanie tempa zbieżności wybranych implementacji metody LBM. Błąd względny został wyrażony w procentach w funkcji liczby iteracji dla wersji CPU oraz GPU dla trzech rozmiarów sieci. Dodatkowo dla rozmiaru L=33 zaznaczone zostały wyniki osobno dla pojedynczej (sp) i podwójnej (dp) precyzji obliczeń.

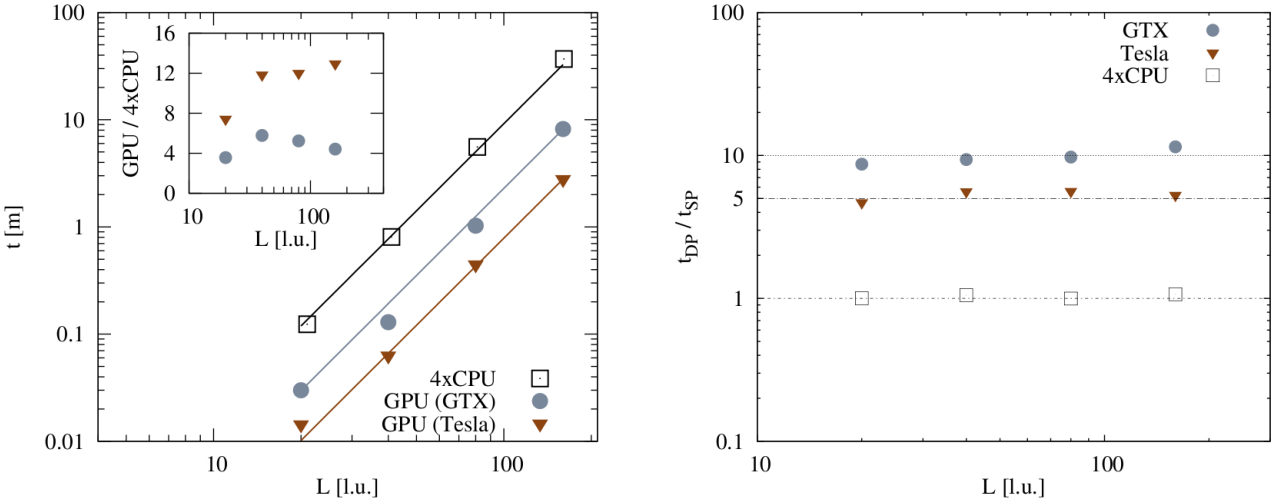
Widać, że minimalna liczba iteracji potrzebna do osiągnięcia stanu stacjonarnego to około 10 000 dla L=33, 25 000 dla L=65 i co najmniej 50 000 dla L=129. Obliczenia kodem Palabos wykazują nieco wolniejszą zbieżność od obliczeń wykonanych kodem Sailfish (wymagają około 25 000 iteracji dla L=33). Widać wyraźną zależność asymptotycznego błędu rozwiązania w funkcji L: błąd jest rzędu 10% dla L=33 i zmniejsza się do 4% dla L=65 i ok. 2% dla L = 129.

Na wykresie 4 wyrysowaliśmy zależność czasu wykonania 5000 iteracji algorytmu LBM w problemie komory trójwymiarowej na GPU i CPU. Przyspieszenia wyrysowane we wstawce do wykresu pokazują wyraźną przewagę rozwiązania GPU z najszybszą kartą Tesla. Dodatkowo porównaliśmy (ryc. 4 prawa) wydajność kodów w wersjach uruchomionych w pojedynczej i podwójnej precyzji. O ile w kodzie uruchomionym na CPU nie ma istotnych różnic, to na GPU wersja działająca w pojedynczej precyzji jest kilkukrotnie szybsza od wersji uruchomionej w podwójnej precyzji obiczeń.

DYSKUSJA

Przeprowadzone obliczenia weryfikacyjne potwierdziły zgodność otrzymanych rozwiązań z wynikami metod spektralnych. Jedną z cech algorytmu LBM jest wolna zbieżność do rozwiązań stacjonarnych [33], drugą – wysokie wymagania odnośnie rozdzielczości sieci obliczeniowej (LBM potrzebuje minimum 4 jednostek sieci do rozwiązania na nich równań Naviera–Stokesa) [21]. Dlatego sprawdziliśmy, czy obserwowane niedokładności w okolicach ścian komory można przypisać niedostatecznej liczbie iteracji użytych w trakcie obliczeń czy też może niewielkiej

rozdzielczości układu. Na ryc. 3. widać wyraźnie, że dla przyjętej wielkości sieci (L=33) liczba iteracji (30 000) jest wystarczająca. Okazało się jednak, że bardzo istotny wpływ na jakość rozwiązania ma rozdzielczość sieci obliczeniowej.



Ryc. 4. (lewa) Czas wykonania algorytmu na CPU i na GPU (w podwójnej precyzji). Linie reprezentują funkcję $f(x) \propto L^{2.7}$, gdzie stałe proporcjonalności zostały wzięte z dofitowania. (prawa) Przyspieszenie obliczeń w pojedynczej precyzji względem precyzji podwójnej. Pozycje linii poziomych wybrane zostały odgórnie dla lepszej orientacji.

Błąd względem rozwiązania wzorcowego wynosił około 10%, gdy rozdzielczości były porównywalne (rzędu 32^3). To bardzo podobny wynik do szacowania błędu między rozwiązaniem LBM a komercyjnym pakietem CFD Ansys zaprezentowanego niedawno w kontekście przepływów biomedycznych [14]. Dla badanego punktu Sailfish i Palabos dają takie same wyniki końcowe, ale dość nieoczekiwanie wynik tego pierwszego dla L=33 (ryc. 3) zbiega szybciej do rozwiązania wzorcowego: Sailfish potrzebował około 10 000 iteracji, aby osiągnąć poziom 10% błędu, Palabos wymagał zaś aż 20 000 kroków.

Wyniki zebrane na wykresie 4 potwierdziły wyraźną przewagę prędkości wykonania kodu pracującego na GPU. Pokazane we wstawce przyspieszenia wynoszą: 4−6 razy dla karty GTX460 i 8−12 razy dla karty Tesla. Dzisiejsze karty graficzne pracują dużo wydajniej w obliczeniach pojedynczej precyzji. Wynik dla karty GeForce wykazuje tu największe przyspieszenie: 12x, natomiast dla CPU przyspieszenia praktycznie nie ma. Maksymalne przyspieszenie GPU względem 4 rdzeni CPU wynosi do 40x dla obliczeń pojedynczej precyzji na karcie Tesla.

Wyniki naszych eksperymentów numerycznych, w połączeniu z wynikami dostępnymi w literaturze, pozwalają oszacować spodziewane przyspieszenie obliczeń biomedycznych po przeniesieniu ich na platformę GPU. Na przykład w [34] autorzy wyznaczyli przepływ przez aortę brzuszną przy pomocy metody LBM pracującej na klasycznym CPU i uzyskali czasy symulacji niestacjonarnej wynoszące 2 godziny na pojedynczym procesorze i 40 minut na czterech węzłach klastra typu Beowulf (złożonego z procesorów AMD 450 Mhz). Biorąc pod uwagę otrzymane przez nas wyniki dla komory o długości L=55 stałych sieci (co z grubsza odpowiada ilości węzłów sieci użytej w [34]) można się spodziewać, że Sailfish wykona podobną symulację w czasie około 100 sekund w pojedynczej i 13 minut w podwójnej precyzji na karcie graficznej GTX460. Analogiczne czasy dla karty Tesla wyniosłyby odpowiednio ok. 57 sekund i 3 minuty. Szacowanie to może być niedokładne (problem, o którym mowa, jest niestacjonarny i charakteryzuje się bardziej skomplikowaną geometrią niż przepływ analizowany w niniejszym opracowaniu, w obu przypadkach zastosowano też inne warunki brzegowe, ponadto symulacje uruchamiane były na węzłach klastra typu Beowulf a nie na pojedynczym procesorze wielordzeniowym), jednak daje ono

pewne wyobrażenie o spodziewanych przyspieszeniach i możliwych aplikacjach tego typu symulacji po przeniesieniu ich na GPU. Jest to szczególnie istotne w kontekście czasów wykonania, o których donoszą autorzy prac badających przepływy medyczne klasycznymi metodami, np. w [5] pojedyncza symulacja przepływu w aorcie brzusznej na CPU trwała od 17 godzin (obliczenia metodą elementów skończonych) do ponad 300 godzin (obliczenia metodą objętości skończonych). Skrócenie tych czasów do maksymalnie kilku-kilkunastu godzin otworzyłoby zupełnie nowe perspektywy zastosowań CFD w diagnostyce medycznej.

PODSUMOWANIE

Przedstawiliśmy wyniki weryfikacyjne oraz czasy wykonania algorytmu LBM w dwóch różnych implementacjach (Sailfish, Palabos). Oba pakiety oprogramowania oferują podobną dokładność obliczeń, różnią się jednak platformą, na której są uruchamiane. Najbardziej zaskakująca jest przewaga wydajności kodu opartego o GPU nad kodem uruchomionym na 4-rdzeniowym procesorze we wszystkich testowanych konfiguracjach (wielkości sieci, rodzaje kart graficznych, precyzja obliczeń). Przewaga ta jest najbardziej wyraźna w pojedynczej precyzji, ale widoczna jest również w obliczeniach w precyzji podwójnej.

Warto pamiętać, że oba pakiety mają pewne unikalne cechy. Dzięki użyciu protokołu MPI Palabos pozwoli na uruchomienie symulacji o dowolnej wielkości (ograniczonej w zasadzie tylko sumaryczną pamięcią klastra obliczeniowego). Z drugiej strony kod Sailfish przeliczy dany problem kilkadziesiąt razy szybciej nawet na komputerze osobistym. Dlatego można się spodziewać, że rozwiązania oparte na procesorach graficznych (lub ogólnie: procesorach tego typu) znajdą zastosowania wszędzie tam, gdzie potrzeba dużej mocy obliczeniowej bez możliwości lub potrzeby instalacji i obsługi ogromnych klastrów obliczeniowych lub superkomputerów (np. w szpitalach). Wyobraźmy sobie lekarza, który siedząc w swoim gabinecie, pobiera z serwera dane (np. skany tomograficzne) swojego pacjenta, wybiera interesujący organ, a następnie uruchamia symulację, np. przepływu krwi, w interesujących go warunkach. Po odczekaniu godziny może obejrzeć charakterystykę transportu i np. rozkład naprężeń na ściankach naczyniowych wybranego organu (gdyby uruchamiał tę samą symulację na, CPU na wyniki musiałby czekać więcej niż dzień).

Należy pamiętać, że użycie metody LBM w praktyce wymaga jeszcze przezwyciężenia szeregu trudności takich jak: poprawne zdefiniowanie problemu i znaczenia rozwiązania numerycznego dla lekarzy, efektywne wygenerowanie danych (siatek) do symulacji, wybór modelu przepływu i warunków brzegowych, implementacja, uruchomienie, a na koniec weryfikacja otrzymanych wyników.

PODZIĘKOWANIA

Publikacja jest efektem realizacji stażu w projekcie Zielony Transfer współfinansowanego ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

BIBLIOGRAFIA

[1] Q. Long, R. Merrifield, G. Yang, X. Y. Xu, P. Kilner, and D. Firmin, *The influence of inflow boundary conditions on intra left ventricle flow predictions*, J Biomech Eng., vol. 125, pp. 922–927, 2003.

[2] Q. Long, R. Merrifield, X. Y. Xu, P. Kilner, D. Firmin, and G. Yang, *Specific computational simulation of left ventricular flow based on magnetic resonance imaging*, J Eng Med., vol. 222, pp. 475–485, 2007.

[3] D. M. McQueen and C. Peskin, *A three-dimensional computer model of the human heart for studying cardiac fluid dynamics*, Comput Graph., vol. 34, no. 56–60, 2000.

[4] S. Khalafvand, E. Ng, and L. Zhong, *CFD simulation of flow through heart: a perspective review*, Computer Methods in Biomechanics and Biomedical Engineering, vol. 14, no. 1, pp. 113–132, 2011.

[5] T. Gohil, R. McGregor, D. Szczerba, K. Burckhardt, K. Muralidhar, and G. Székely, *Simulation of oscillatory flow in an aortic bifurcation using fvm and fem: a comparative study of implementation strategies,* International Journal for Numerical Methods in Fluids, vol. 66, no. 8, pp. 1037–1067, 2011.

[6] R. McGregor, D. Szczerba, and G. Székely, *A multiphysics simulation of a healthy and a diseased abdominal aorta,* in *Proceedings of Medical Image Computing and Computer-Assisted Intervention*, pp. 227–234, 2007.
[7] W. A. Wall and T. Rabczuk, *Fluid–structure interaction in lower airways of CT-based lung geometries*, International Journal for Numerical Methods in Fluids, vol. 57, no. 5, pp. 653–675, 2008.
[8] G. Stamatakos, V. Antipas, and N. Uzunoglu, *A spatiotemporal, patient individualized simulation model of solid tumor response to chemotherapy in vivo: the paradigm of glioblastoma multiforme treated by temozolomide,* IEEE Trans Biomed Eng., vol. 53, no. 8, pp. 1467–77, 2006.
[9] K. Borkenstein, S. Levegrun, and P. Peschke, *Modeling and computer simulations of tumor growth and tumor response to modeling and computer simulations of tumor growth and tumor response to modeling and computer simulations of tumor growth and tumor response to radiotherapy*, Radiation Research, vol. 162, no. 1, pp. 71–83, 2004.
[10] G. S. Stamatakos, E. I. Zacharaki, N. K. Uzunoglu, and K. S. Nikita, *Tumor growth and response to irradiation in vitro: A tumor growth and response to irradiation in vitro: A technologically advanced simulation model*, Int. J. Radiat. Oncol. Biol. Phys., vol. 51, no. 3, pp. 240–241, 2001.
[11] J. P. Sinek, S. Sanga, X. Zheng, H. B. Frieboes, M. Ferrari, and V. Cristini, *Predicting drug pharmacokinetics and effect in vascularized tumors using computer simulation*, J. Math. Biol., vol. 58, no. 4-5, pp. 485–510, 2009.
[12] Z. Malecha, Ł. Mirosław, T. Tomczak, Z. Koza, M. Matyka, W. Tarnawski, and D. Szczerba, *Gpu-based simulation of 3d blood flow in abdominal aorta using openfoam*, Archives of Mechanics, vol. 63, no. 2, pp. 137–161, 2011.
[13] J. Bernsdorf and D. Wang, *Non-Newtonian blood flow simulation in cerebral aneurysms*, Computers & Mathematics with Applications, vol. 58, pp. 1024–9, Sept. 2009.
[14] B. Chopard, D. Lagrava, O. Malaspinas, R. Ouared, J. Latt, K.-O. Lovblad, and V. Pereira-Mendes, *A lattice boltzmann modelling of blood flow in cerebral aneurysm* in V European Conference on Computational Fluid Dynamics (J. C. F. Pereira and A. Sequeira, eds.), pp. 330—331, June 2010.
[15] U. Frisch, B. Hasslacher, and Y. Pomeau, *Lattice-gas automata for the Navier-Stokes equation*, Phys. Rev. Lett., vol. 56, pp. 1505–1508, Apr 1986.
[16] G. R. McNamara and G. Zanetti, *Use of the Boltzmann equation to simulate lattice-gas automata*, Phys. Rev. Lett., vol. 61, pp. 2332–2335, Nov 1988.
[17] X. He and L.-S. Luo, *Lattice boltzmann model for the incompressible Navier Stokes equation*, Journal of Statistical Physics, vol. 88, pp. 927– 944, Aug. 1997.
[18] S. Ansumali and I. V. Karlin, *Consistent lattice Boltzmann method*, Phys. Rev. Lett., vol. 95, p. 260605, Dec 2005.
[19] S. S. Chikatamarla, S. Ansumali, and I. V. Karlin, *Entropic Lattice Boltzmann models for hydrodynamics in three dimensions*," Phys. Rev. Lett., vol. 97, p. 010201, Jul 2006.
[20] D. d'Humieres, I. Ginzburg, M. Krafczyk, P. Lallemand, and L. S. Luo, *Multiple-relaxation-time lattice Boltzmann models in three dimensions*, Philosophical Transactions: Mathematical, Physical and Engineering Sciences, vol. 360, no. 1792, pp. 437+, 2002.
[21] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond* (Numerical Mathematics and Scientific Computation). Numerical mathematics and scientific computation, Oxford University Press, USA, 2001.
[22] J. Latt, Hydrodynamic limit of lattice Boltzmann equations. PhD thesis, Univ. Genève, 1997.
[23] C. K. Aidun and J. R. Clausen, *Lattice-Boltzmann Method for Complex Flows,* Annual Review of Fluid Mechanics, vol. 42, no. 1, pp. Annual Reviews—472, 2011.
[24] J. Bernsdorf, *Simulation of Complex Flows and Multi-Physics with the Lattice-Boltzmann Method*. PhD thesis, University of Amsterdam, 2008.
[25] internet: *http://www.numhpc.org/openlb/*, 2011.
[26] internet: *http://www.palabos.org/*, 2011.
[27] O. B. W. Degruyter, A. Burgisser and O. Malaspinas, *Synchrotron X-ray microtomography and lattice boltzmann simulations of gas flow through volcanic pumices*," Geosphere, vol. 6, no. 5, pp. 470–481, 2010.
[28] internet: *http://sailfish.us.edu.pl/*, 2011.
[29] C. Feichtinger, S. Donath, H. Kötler, J. Götz, and U. Rüde, *Walberla: HPC software design for computational engineering simulations*," Tech. Rep., Friedrich-Alexander-Universität Erlagen-Nürnberg, 2010.
[30] B. Chareyre, A. Cortis, E. Catalano, and E. Barthélemy, Pore-scale modeling of viscous flow and induced forces in dense sphere packings, 2011: arXiv:1105.0297v2
[31] S. Albensoeder and H. C. Kuhlmann, *Accurate three-dimensional lid- driven cavity flow*, Journal of Computational Physics, vol. 206, pp. 536– 558, July 2005.
[32] O. Botella and R. Peyret, *Benchmark spectral results on the lid-driven cavity flow*, Computers & Fluids, vol. 27, no. 4, pp. 421 – 433, 1998.
[33] M. Bernaschi, S. Succi, and H. Chen, *Accelerated lattice boltzmann schemes for steady-state flow simulations* J. Sci. Comput., vol. 16, no. 2, pp. 135–144, 2001.
[34] Artoli, A. and Hoekstra, A. and Sloot, P., *Mesoscopic simulations of systolic flow in the human abdominal aorta*, J. Biomech., vol. 39, pp. 873+, 2006.

SUMMARY

PERFORMANCE OF OPEN-SOURCE IMPLEMENTATIONS
OF THE LATTICE BOLTZMANN METHODS ON CPU AND GPU

The Lattice Boltzmann Method (LBM) is a simulation method in the field of computational fluid dynamics . The goal of this work is to investigate two open source implementations of the LBM: Palabos and Sailfish. The former runs on the classical multicore CPU and the latter runs on a single Graphical Processing Unit (GPU). We discuss how the precision, type of an application, physical conditions and processor architecture influence the performance of both codes. In each case the code run on GPU is faster than its counterpart running on a multicore CPU. We report GPU versus CPU speedup of 3x-12x in double precision and 24x-42x in single precision computer arithmetic.