

Praca Przejściowa Inżynierska
Analiza skalowalności kodu TCLB opartego o
metodę Lattice Boltzmann

Rybski Arkadiusz
Mechanika i Projektowanie Maszyn
numer indeksu 292784

2020

Spis treści

1 Wprowadzenie	3
1.1 Cel pracy	3
1.2 Metoda Lattice Boltzmann	3
1.2.1 Rodzaje krat	3
1.2.2 Algorytm	5
1.2.3 Rodzaje kernela	5
1.3 Skalowalność kodu	6
1.3.1 Silne skalowanie	7
1.3.2 Słabe skalowanie	8
1.3.3 Skalowanie superliniowe	9
2 Analiza	9
2.1 Wprowadzenie	9
2.2 Opis klastrów obliczeniowych	10
2.2.1 Prometheus	10
2.2.2 Rysy	10
2.3 Badana geometria	10
3 Wyniki	12
3.1 Prezentacja wyników	12
3.1.1 Silne Skalowanie	12
3.1.2 Słabe Skalowanie	14
3.1.3 Wykresy przepływ informacji	18
3.1.4 Wydajność GPU	25
3.2 Analiza wyników	26

1 Wprowadzenie

1.1 Cel pracy

Celem pracy było zbadanie skalowalności kodu obliczeniowego metody Lattice Boltzmann w celu określenia jej efektywności. Dzięki zrealizowanym badaniom możemy określić optymalne wykorzystanie zasobów obliczeniowych.

1.2 Metoda Lattice Boltzmann

Metoda Lattice Boltzmann jest metodą numeryczną służącą do rozwiązywania równań z zakresu mechaniki płynów. Metoda kratowa Boltzmanna oparta jest na równaniu Boltzmanna([Rownanie 1](#)):

$$\frac{\partial f}{\partial t} + \xi_\beta \frac{\partial f}{\partial x_\beta} + \frac{F_\beta}{\rho} \frac{\partial f}{\partial \xi_\beta} = \Omega(f) \quad (1)$$

gdzie:

$f(x, \xi, t)$ - oznacza funkcję rozkładu

x - oznacza położenie

ξ - oznacza prędkości cząstek

t - oznacza czas

$\Omega(f)$ - oznacza operator kolizji([1.2.3](#)).

Można pokazać, że rozwiązania mezoskopowych równań Boltzmanna zbiega się do równań Naviera Stokesa. Niestety nie rozwiązuje to problemu analitycznego rozwiązania równania. Natomiast okazuje się, iż mimo skomplikowanej formy, równanie Boltzmanna w prosty sposób można zaimplementować. W ten sposób możemy otrzymać równanie kratowe Boltzmanna ([Rownanie 2](#)):

$$f_i(x + c_i \Delta t, t + \Delta t) = f_i(x, t) + \Omega(x, t) \quad (2)$$

gdzie:

$c_i = (c_{ix}, c_{iy}, c_{iz})$ - oznacza prędkość cząstki w i-tym węźle.

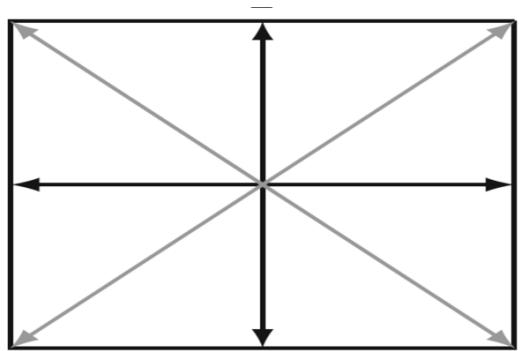
1.2.1 Rodzaje krat

Ze względu na to, że funkcja dystrybucji jest zależna nie tylko od czasu, położenia ale też i prędkości do implementacji potrzebujemy siatki zawierającej nie tylko położenie geometryczne. Wymagana jest dyskretyzacja czasu, przestrzeni, ale także prędkości. W literaturze przyjęto następujące oznaczenia krat:

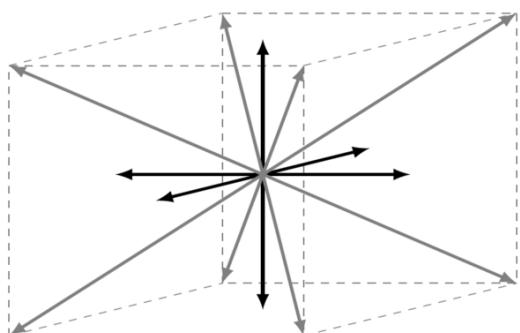
$$D_n Q_m$$

gdzie n oznacza ilość wymiarów, natomiast m oznacza ilość możliwych kierunków prędkości.

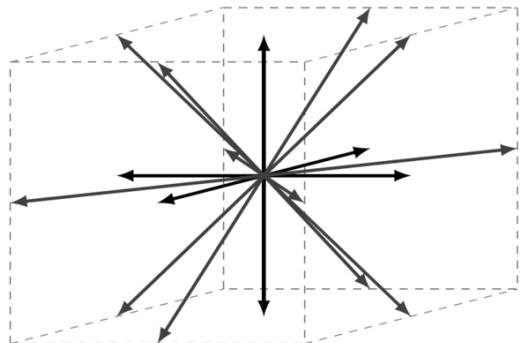
Przykładowe kraty zostały zamieszczone na poniższych obrazkach ([2], [4]).



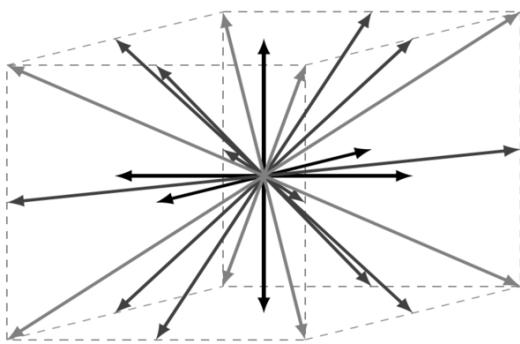
(a) D2Q9



(b) D3Q15



(c) D3Q19



(d) D3Q27

Rysunek 1: Przykładowe kraty

1.2.2 Algorytm

Zasada działania metody kratowej Boltzmanna oparta jest na dwóch fazach: propagacji i kolizji. Poniższe równania będą przedstawione dla operatora kolizji BGK ([4]).

Faza kolizji

$$f_i^*(x, t) = f_i(x, t) - \frac{\Delta t}{\tau} (f_i(x, t) - f_i^{eq}(x, t)) \quad (3)$$

Faza propagacji

$$f_i(x + c_i \Delta t, t + \Delta t) = f_i^*(x, t) \quad (4)$$

Całość mechanizmu można podsumować w następujących krokach:

1. Wybór lokalizacji
2. Rejestracja informacji o nadchodzących cząsteczkach
3. Kolizja
4. Dystrybucja po kolizji
5. Wybór kolejnej lokalizacji

1.2.3 Rodzaje kernela

Przedstawiony w powyższym algorytmie operator kolizji *BGK* (*Bhatnagar-Gross-Krook*) to jeden z wielu możliwych operatorów kolizji. Inne z nich to *MRT* (*Multiple Relaxation Time*), *Central Moment* [1] czy *Cumulant Collision Operator* [3].

Operator kolizji musi spełniać równania zachowania masy ([Równanie 5](#)), momentów ([Równanie 6](#)), energii ([Równanie 7](#)), a także zasadę zachowania entropii[4].

$$\int \Omega(f) d^3\xi = 0 \quad (5)$$

$$\int \Omega(f) \xi d^3\xi = 0 \quad (6)$$

$$\int \Omega(f) \xi^2 d^3\xi = 0 \quad (7)$$

Operator kolizji może być realizowany na wiele sposobów, dwa z nich zamieszczone zostały poniżej.

BGK operator [4]

$$\Omega(f) = -\frac{\Delta t}{\tau} (f_i(x, t) - f_i^{eq}(x, t)) \quad (8)$$

gdzie:

τ - oznacza stałą czasową, nazywaną czasem relaksacji

f^{eq} - oznacza funkcję rozkładu w stanie równowagi (*equilibrium*).

Multiple relaxation time colission operator [4]

$$\Omega(f) = -M^{-1} S M [f(x, t) - f^{eq}(x, t)] \Delta t \quad (9)$$

gdzie:

M - oznacza macierz transformacji

S - oznacza macierz relaksacji (kolizji).

1.3 Skalowalność kodu

Dzięki nieustannemu rozwojowi technicznemu współcześnie jesteśmy w stanie rozwiązywać większe problemy obliczeniowe za pomocą wielu procesorów, co zdecydowanie skraca czas wykonywania obliczeń. Skalowalność określa nam efektywność kodu w przypadku użycia zwiększych zasobów komputerowych. W celu zbadania efektywności obliczeń równoległych wprowadźmy wielkość zwana dalej *przyspieszeniem* (z ang. *speedup*):

$$speedup = \frac{t_1}{t_N}$$

gdzie:

t_1 - oznacza czas wykonania procesu przy użyciu 1-ego procesora

t_N - oznacza czas wykonania procesu przy użyciu N procesorów.

W idealnym przypadku wykres $speedup(N)$ byłby wykresem liniowym.

Drugą wielkością, która wprowadzimy będzie tzw. *skalowane przyspieszenie* (z ang. *scaled speedup*):

$$scaled\ speedup = \frac{t_1}{\frac{t_N}{N}}$$

gdzie:

t_1 - oznacza czas wykonywania procesu przy użyciu 1 procesora

t_N - oznacza czas wykonywania procesu przy użyciu N procesorów, pod warunkiem stałej wielkości $\frac{work}{processor}$.

Przyspieszenie jest limitowane przez część kodu obliczeniowego, której nie jesteśmy w stanie zrównoleglić. Istnieją dwa główne podejścia w zakresie badania skalalności programów: silne i słabe skalowanie.

1.3.1 Silne skalowanie

Idea silnego skalowania jest prosta: przy zachowaniu stałego rozmiaru programu zwiększamy ilość procesorów pracujących nad jego rozwiązaniem. *Przyspieszenie* jest limitowane prawem Amdahl'a:

$$speedup \leq \frac{1}{s + \frac{p}{N}} < \frac{1}{s}$$

gdzie:

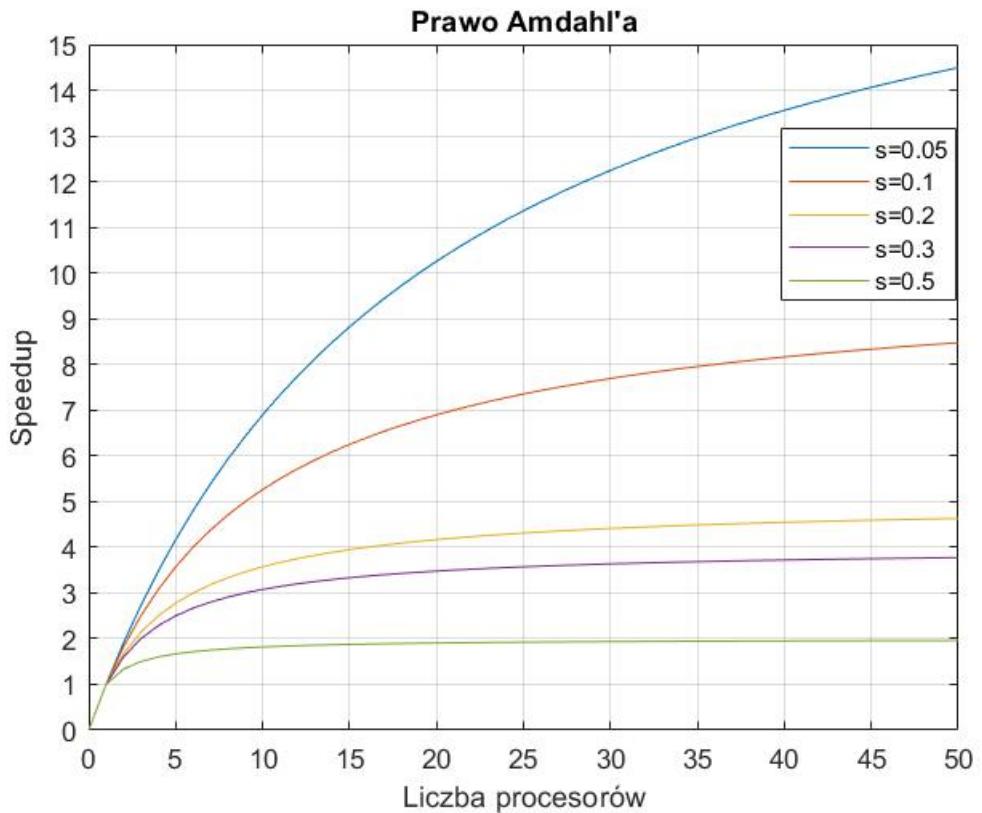
s - oznacza część kodu, która nie może zostać zrównoleglona

p - część kodu zrównoleglona

N - oznacza liczbę procesorów.

Uwaga: $s + p = 1$.

Przykładowo jeśli 10% kodu obliczeniowego nie nadaje się do zrównoleglenia to maksymalnie możemy uzyskać 10-krotne przyspieszenie procesu.



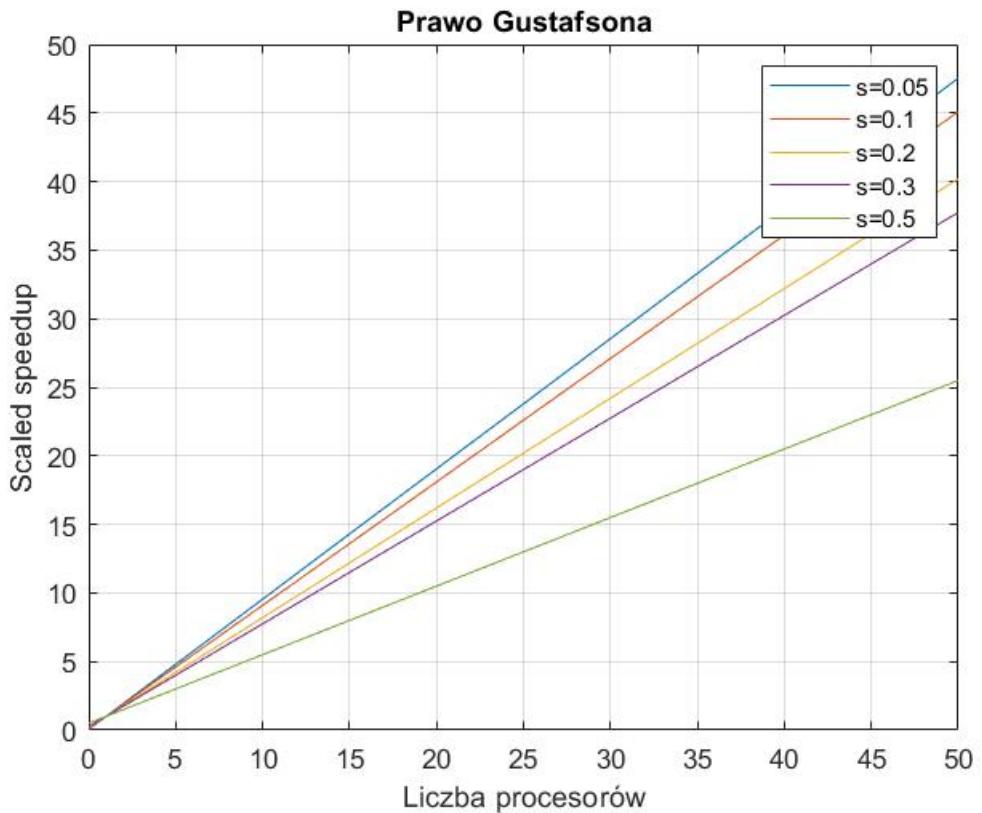
Rysunek 2: Prawo Amdahl'a

1.3.2 Słabe skalowanie

Drugim podejściem do badania skalowalności jest słabe skalowanie(z ang. *weak scaling*). W tym przypadku zwiększany jest równocześnie rozmiar problemu jak i ilość procesorów pracujących nad jego rozwiązaniem. *Przyspieszenie skalowane* limitowane jest prawem Gustafson'a:

$$\text{scaled speedup} \leq s + pN$$

oznaczenia jak wyżej.



Rysunek 3: Prawo Gustafson'a

1.3.3 Skalowanie superliniowe

Należy także wspomnieć o zjawisku zwanym *superliniowym skalowaniem*. Jest to zjawisko, w którym w miarę zwiększania liczby procesorów czas wykonywania zmniejsza się bardziej niż liniowo. Istnieją różne wyjaśnienia tego zjawiska. Jednym z nich jest dostęp do pamięci cache. Innym wyjaśnieniem zjawiska superliniowego skalowania może być sup optymalny algorytm sekwencyjny, np. stosowanie algorytmu drzewa binarnego. Gdy koszt pojedynczego algorytmu wynosi $O(n^2)$ to w przypadku podzielenia tego procesu na dwa procesory koszt wyniesie $O((0.5n)^2) = O(0.25n^2)$.

2 Analiza

2.1 Wprowadzenie

Kluczowym parametrem metod numerycznych jest stosunek ilości informacji przekazywanych między procesorami do całej objętości problemu. W metodzie Lattice Boltzmann informacja jest przekazywana na granicy styku

subdomen przypisanych do kart GPU. Dlatego w tym przypadku liczy się stosunek powierzchni przepływu informacji do objętości całego problemu obliczeniowego.

Przeprowadzone zostały badania skalowalności kodu TCLB metody Lattice Boltzmann oraz badania wpływu kształtu domeny obliczeniowej na szybkość wykonywanej symulacji.

Wszystkie symulacje zostały przeprowadzone w oparciu o solver TCLB ([Solver TCLB](#)).

2.2 Opis klastrów obliczeniowych

2.2.1 Prometheus

Typ

Klaster GPU

Architektura

Intel Xeon (Haswell / Skylake)

Liczba rdzeni obliczeniowych

53604

Liczba kart GPGPU

144 (Nvidia Tesla K40 XL)

2.2.2 Rysy

Typ

Klaster GPU

Architektura

Intel Skylake NVIDIA Volta

Liczba rdzeni obliczeniowych

216

Liczba kart GPGPU

24 GPU (Nvidia V100 32GB)

2.3 Badana geometria

Badania zostały oparte o symulacje przepływu z wymianą ciepła wokół walca. Przestrzeń obliczeniowa została zdefiniowana następująco:

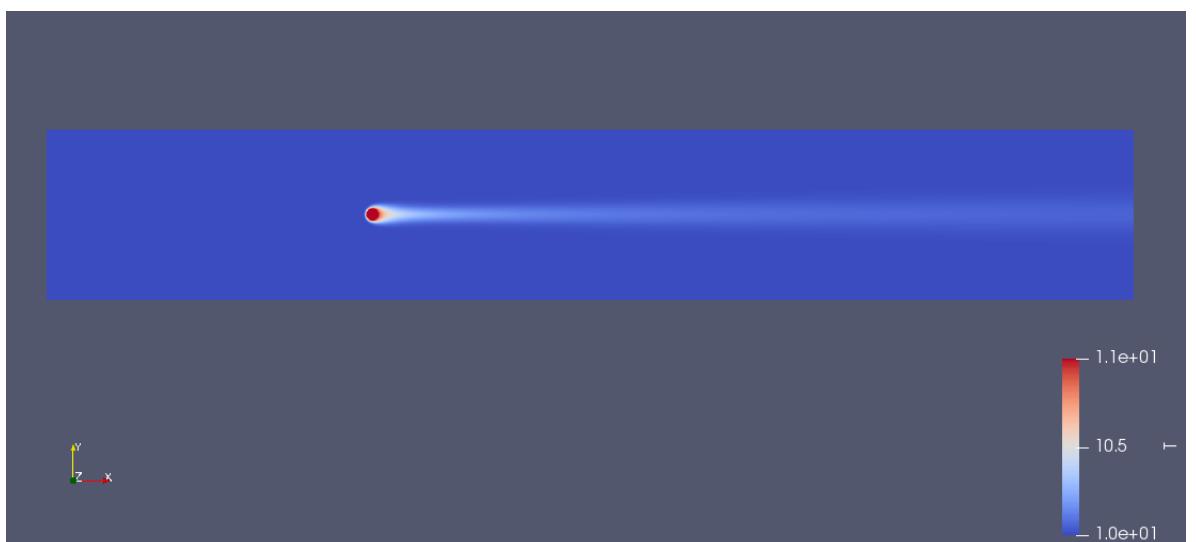
- długość przestrzeni na X - Ln , $n = 1, 2, 4, 8$

- długość przestrzeni na Y - Hm , $m = 1, 2, 4, 8, 16, 32, 64, 128$
- długość przestrzeni na Z - 3
- średnica cylindra $d = 3m$, m jw.
- położenie środka cylindra
 - $x_{cylindra} = 0.3Ln$
 - $y_{cylindra} = 0.5Hm$

gdzie środek układu współrzędnych to lewy dolny róg prostokąta widocznego na załączonym zdjęciu([Rysunek 4](#)).

Zostały dobrane dwa bazowe rozmiary siatki:

- $L = 2816$, $H = 55$
- $L = 4096$, $H = 80$



Rysunek 4: Przepływ wokół walca

Uwagi

W kodzie TCLB domena obliczeniowa jest dzielona po osi Y.

Wyjątkowe spłaszczone domeny obliczeniowe zostały dobrane w celu zbierania wpływu powierzchni przepływu informacji do objętości całkowitej problemu ([3.1.3](#)).

Krata W symulacji użyto trójwymiarowej kraty typu D3Q27 ([Rysunek 1d](#)) dla termiki oraz dla D3Q27([Rysunek 1d](#)) dla hydrodynamiki.

Kernel Zastosowane zostały dwa kernele:

- kernel hydrodynamiczny - *Cumulant Collision Operator* [3]
- kernel populacji termicznych - *Central Moment Operator* [1]

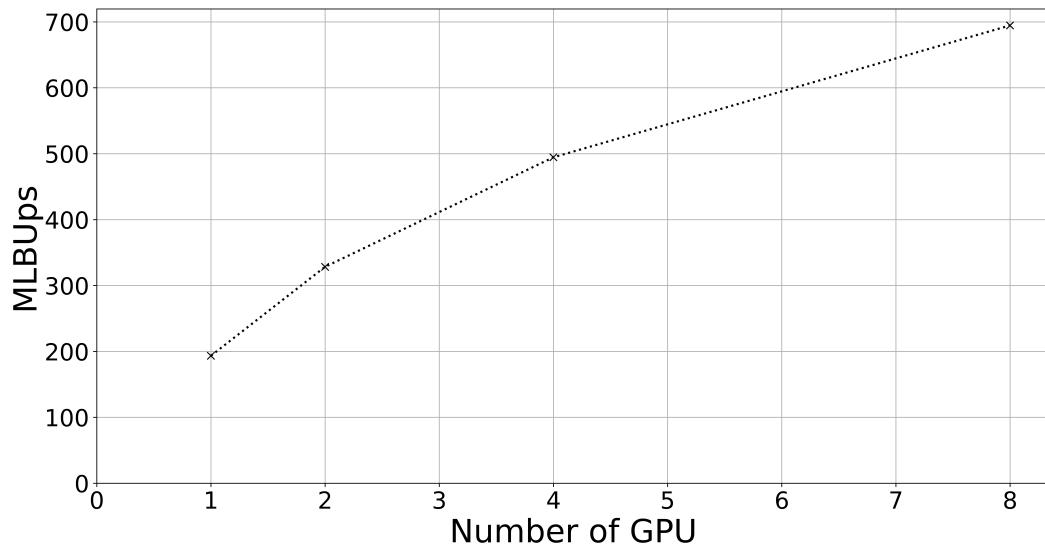
3 Wyniki

3.1 Prezentacja wyników

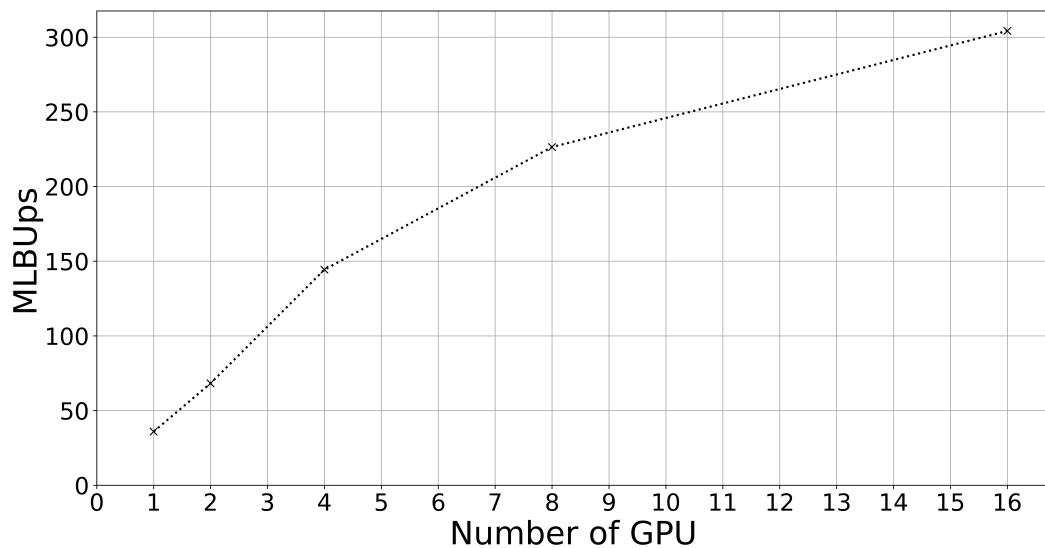
Wyniki symulacji zostały przedstawione poniżej. Na osi Y przedstawiona jest szybkość symulacji w jednostkach MLBUps (*Million Lattice Boltzmann Updates Per Second*). Szybkość symulacji o wartości 1MLBUps oznacza, że siatka o rozmiarze miliona elementów jest aktualizowana raz na sekundę. Na osi X dla silnego skalowania([3.1.1](#)), oraz słabego skalowania([3.1.2](#)) znajduje się liczba użytych do symulacji kart GPU. Natomiast dla wykresów dot. wpływu kształtu domeny na szybkość obliczeń([3.1.3](#)) na osi X znajduje się stosunek powierzchni przepływu informacji do objętości domeny na kartę(wyjaśnienie: [3.1.3](#))

3.1.1 Silne Skalowanie

Wykresy dot. silnego skalowania zostały stworzone w wyniku symulacji powyżej opisanej geometrii z siatką o rozmiarze $4096 \times 640 \times 3$, która zajmowała $\sim 10GB$ pamięci karty GPU.



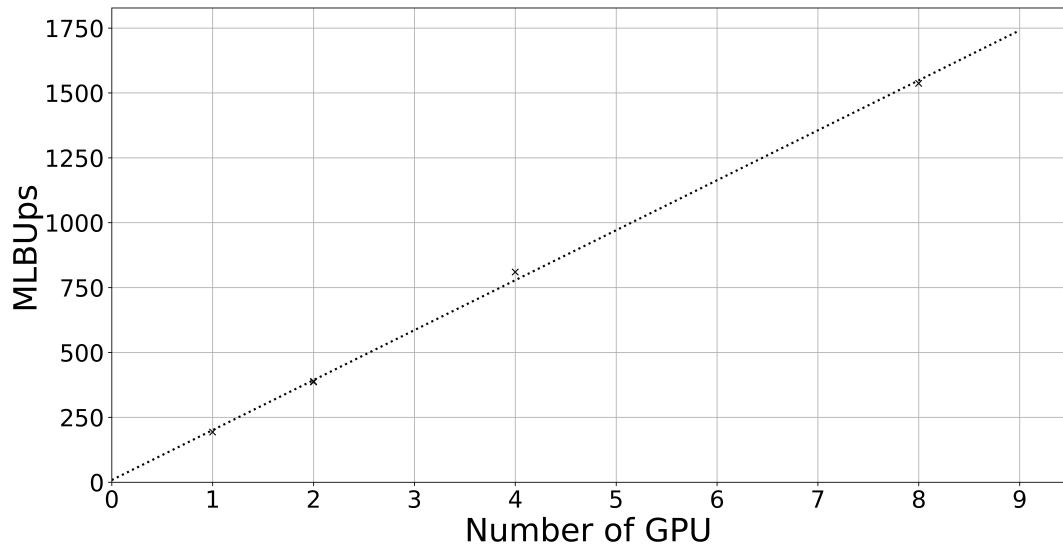
(a) Rysy



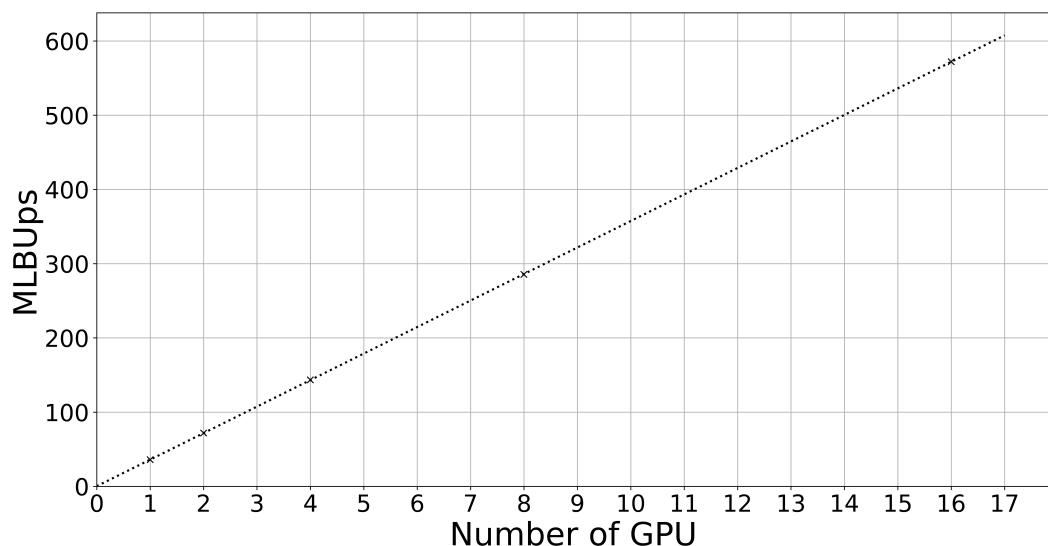
(b) Prometheus

3.1.2 Słabe Skalowanie

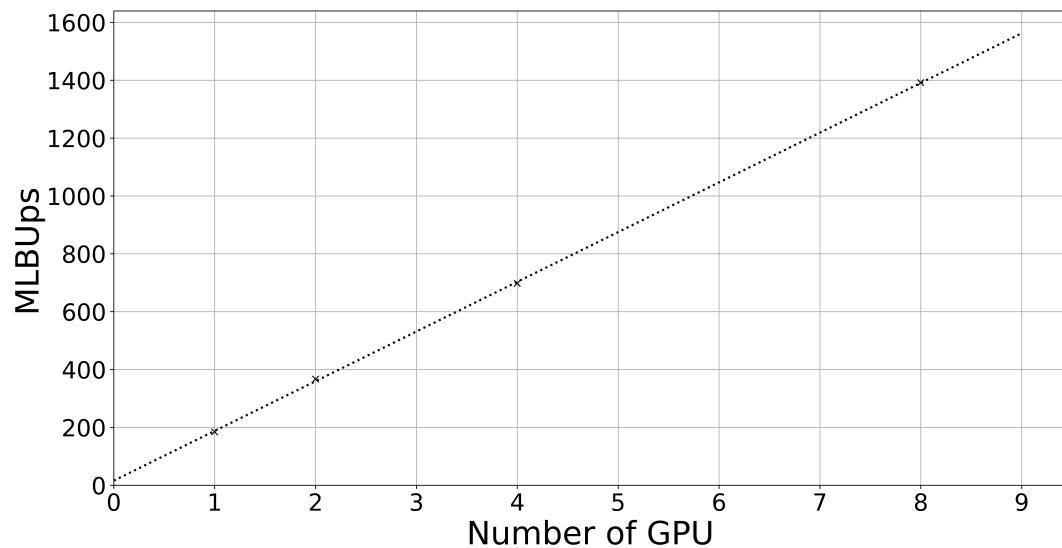
Przedstawione zostały wykresy dla skrajnych konfiguracji siatek(tzn. dla najmniejszego oraz największego stosunku długości domeny obliczeniowej na osi X do długości domeny obliczeniowej na osi Y), oraz dla dwóch bazowych siatek, które zajmowały $\sim 5GB$ oraz $\sim 10GB$ pamięci karty.



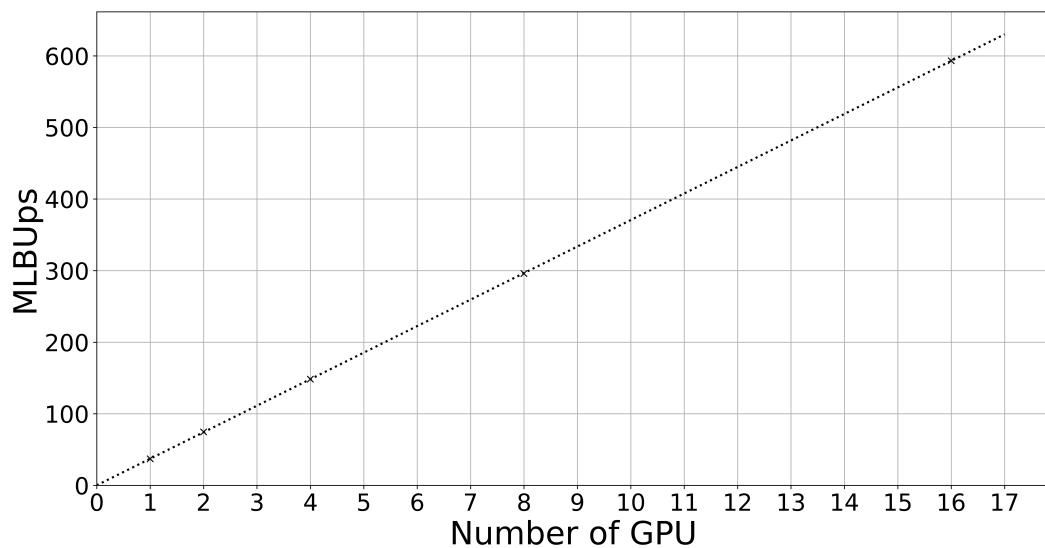
(a) Rysy: $\{4096, 4096, 4096, 4096\} \times \{640, 1280, 2560, 5120\} \times \{3, 3, 3, 3\}$



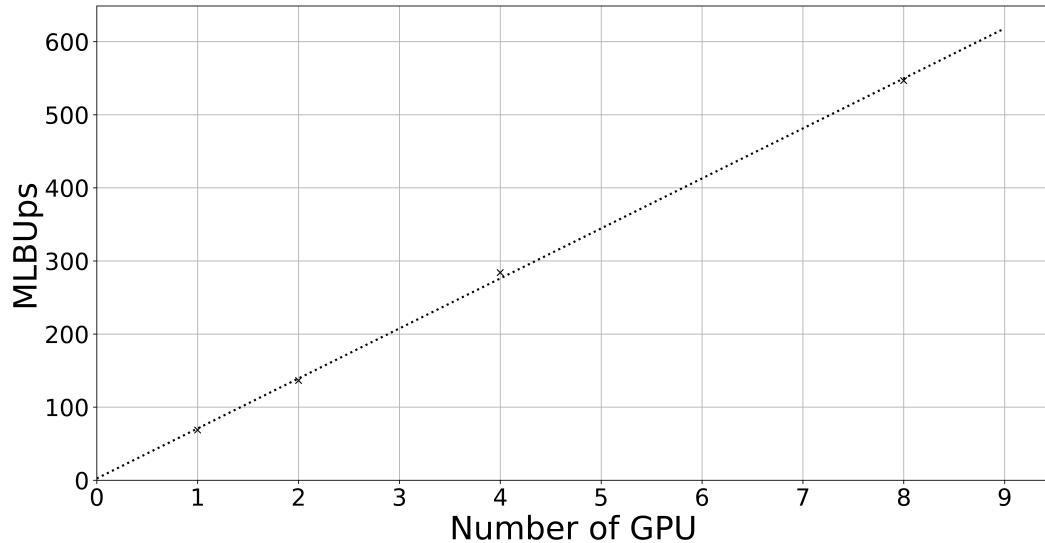
(b) Prometheus: $\{4096, 4096, 4096, 4096, 4096\} \times \{640, 1280, 2560, 5120, 10280\} \times \{3, 3, 3, 3, 3\}$



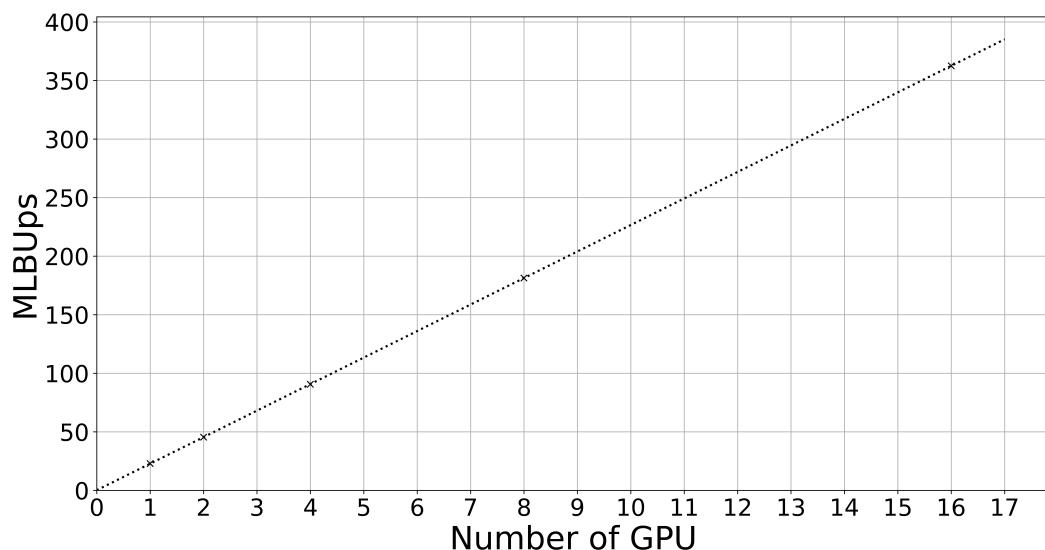
(c) Rysy: {2816, 2816, 2816, 2816}x{440, 880, 1760, 3520}x{3, 3, 3, 3}



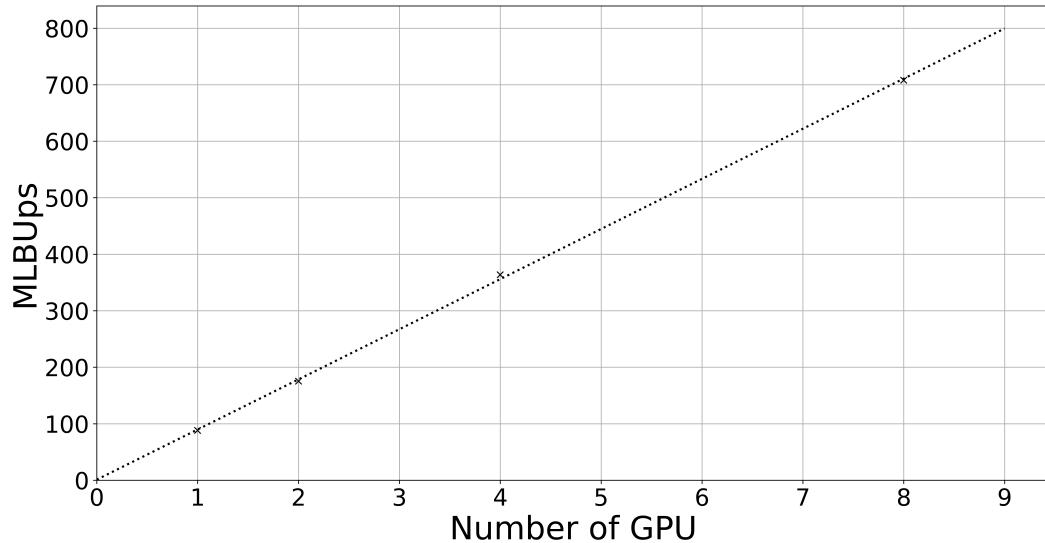
(d) Prometheus: {2816, 2816, 2816, 2816, 2816}x{440, 880, 1760, 3520, 7040}x{3, 3, 3, 3, 3}



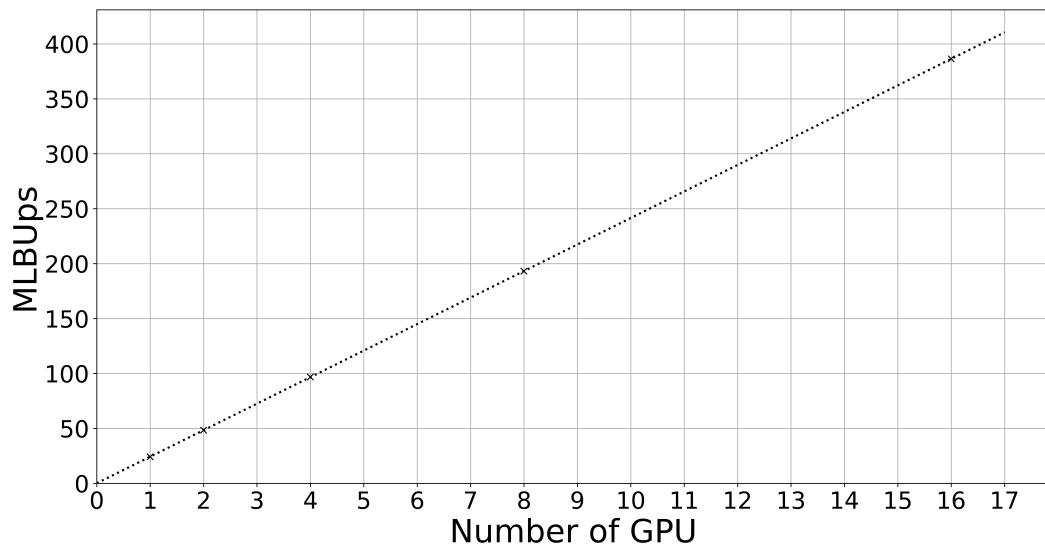
(e) Rysy: {22528, 22528, 22528, 22528}x{55, 110, 220, 440}x{3, 3, 3, 3}



(f) Prometheus: {22528, 22528, 22528, 22528, 22528}x{55, 110, 220, 440, 880}x{3, 3, 3, 3, 3}



(g) Rysy: $\{32768, 32768, 32768, 32768\} \times \{80, 160, 320, 640\} \times \{3, 3, 3, 3\}$



(h) Prometheus: $\{32768, 32768, 32768, 32768, 32768\} \times \{80, 160, 320, 640, 1280\} \times \{3, 3, 3, 3, 3\}$

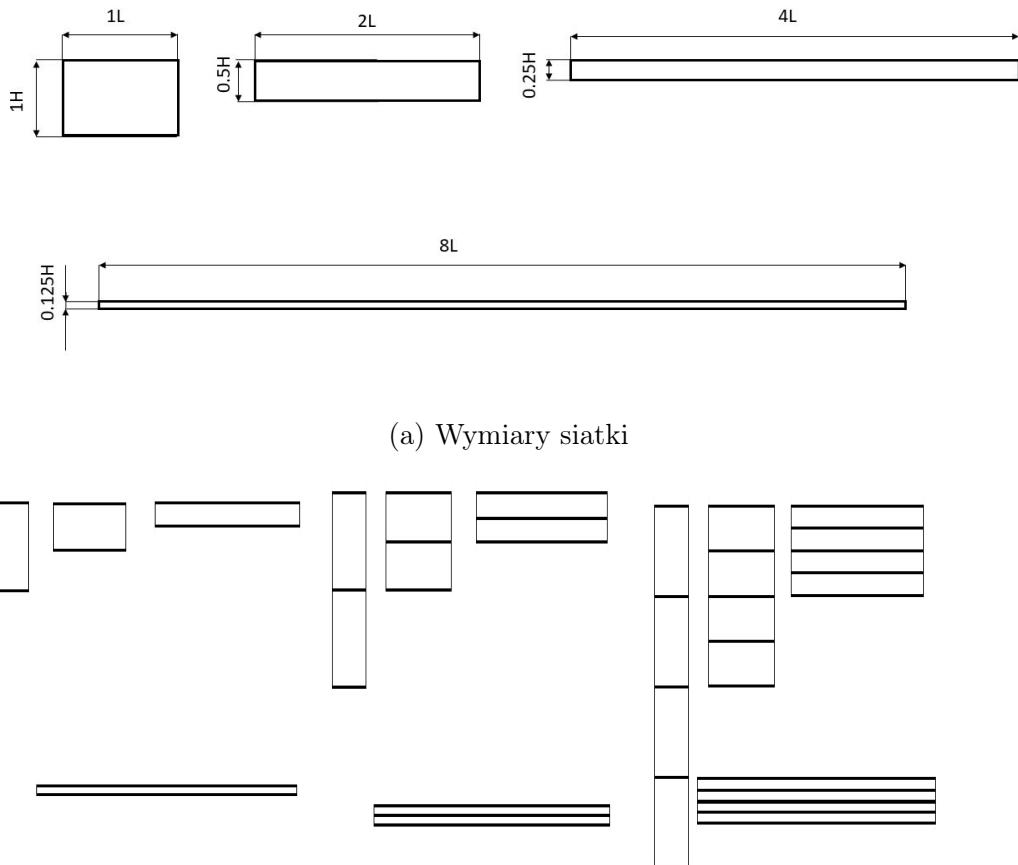
Rysunek 6: Weak scaling

3.1.3 Wykresy przepływów informacji

Kluczową rolę w skalowalności kodu obliczeniowego metody Lattice Boltzmann odgrywa stosunek ilości tzw. *ghost cells* (węzły siatki obliczeniowej na granicy obszaru obliczeniowego znajdującego się na karcie) do objętości domeny na kartę.

W celu zbadania wpływu stosunku powierzchni przepływu informacji do objętości domeny obliczeniowej na kartę zastosowano schemat obliczeń widoczny na poniższej ilustracji ([Rysunek 7](#)). Obliczenia na 8 i 16 kartach przeprowadzono analogicznie.

Symulacje przeprowadzono na dwóch bazowych rozmiarach siatki, które zajmowały odpowiednio $\sim 5GB(L = 2816, H = 440)$ oraz $\sim 10GB(L = 4096, H = 640)$ pamięci karty.



Rysunek 7: Schemat obliczeń

Dalej przyjęto następująco oznaczenia:

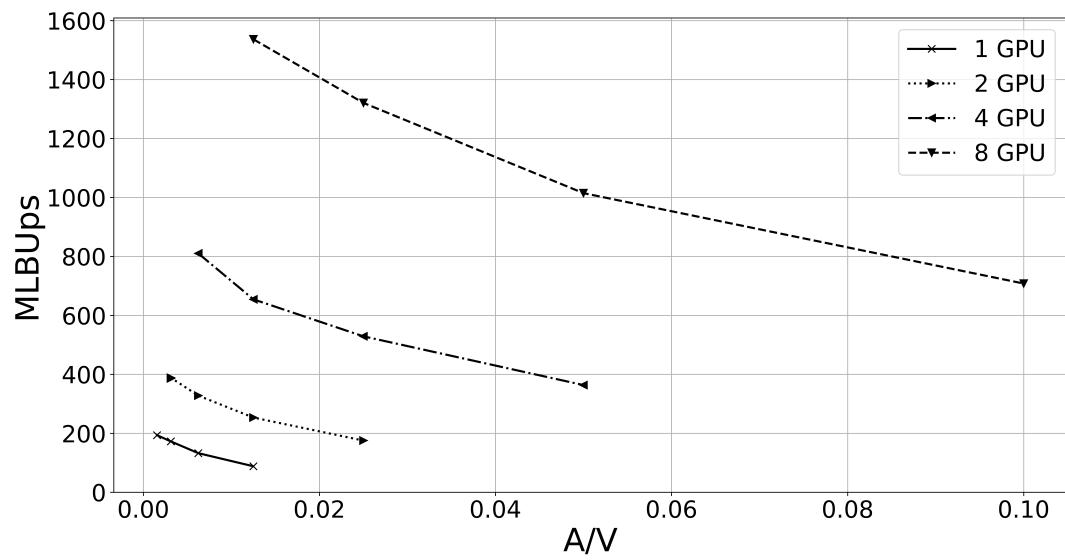
$A = XZn$ - oznacza powierzchnię przepływu informacji

$V = \frac{XYZ}{n}$ - oznacza objętość domeny obliczeniowej na 1 kartę

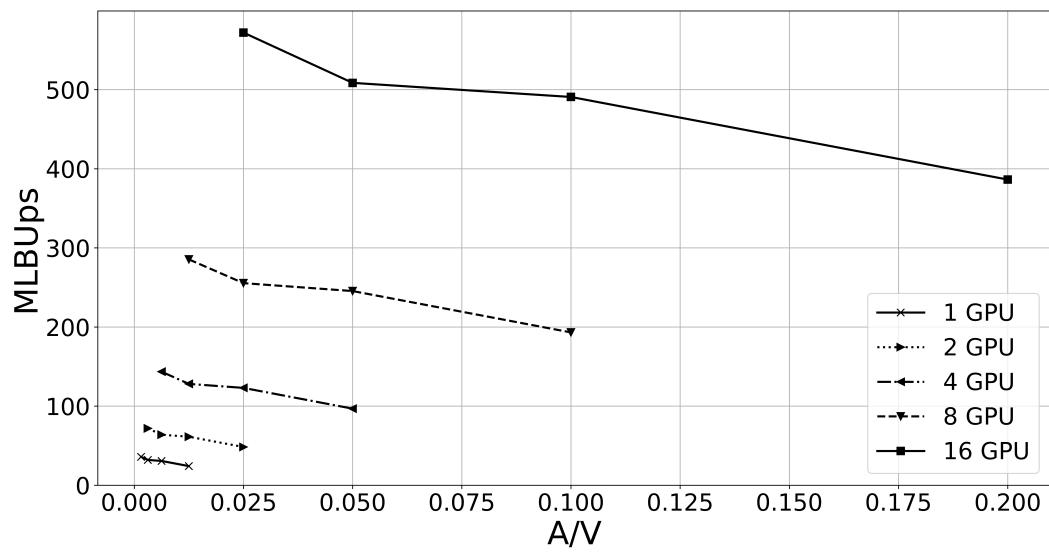
X, Y, Z - oznaczają kolejno całkowitą długość domeny obliczeniowej na osiach X, Y, Z

n - oznacza liczbę kart GPU.

Poniżej przedstawiono wykresy oraz tabele mające na celu zbadać te zależności.

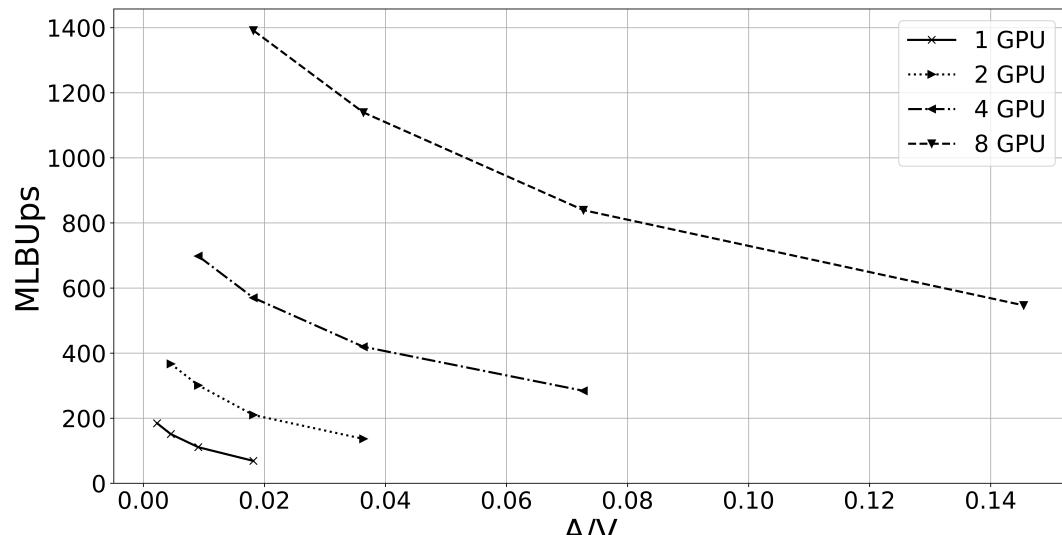


(a) Rysy

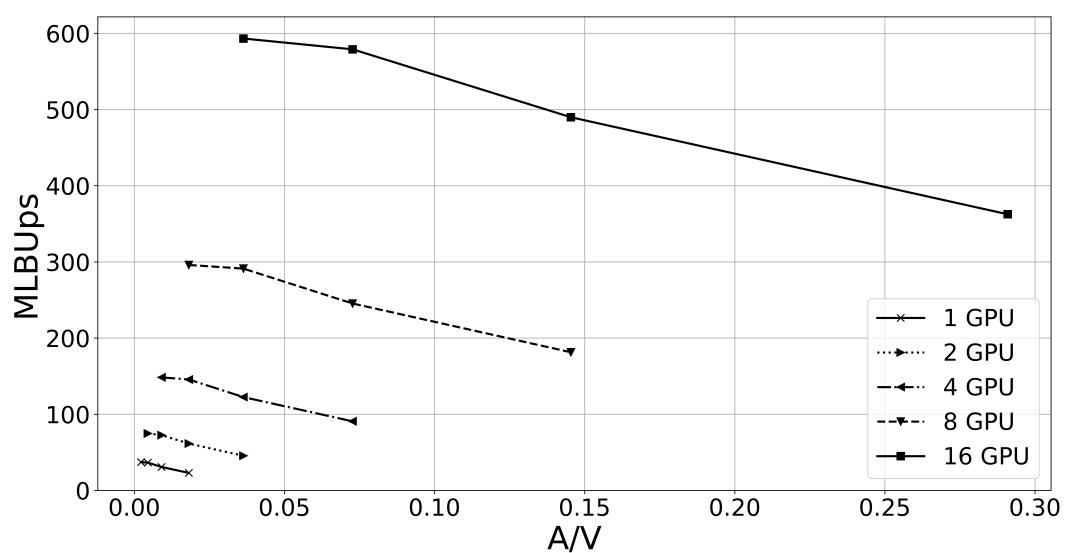


(b) Prometheus

Rysunek 8: A/V plots 10GB load



(a) Rysy



(b) Prometheus

Rysunek 9: A/V plots 5GB load

Tablica 1: Siatka $\sim 10GB$

A	V	n GPU	Speed[MLBUpS]	
			Rysy	Prometheus
12288	7864320	1	193.4	35.9
24576	7864320	1	172.4	32.0
49152	7864320	1	132.5	30.7
98304	7864320	1	88.1	24.2
24576	15728640	2	388.4	71.9
49152	15728640	2	327.7	63.8
98304	15728640	2	253.4	61.3
196608	15728640	2	175.3	48.4
393216	31457280	4	363.5	96.8
98304	31457280	4	654.4	123.0
49152	31457280	4	810.0	128.1
196608	31457280	4	528.9	143.5
196608	62914560	8	1320.7	255.4
786432	62914560	8	708.0	193.2
98304	62914560	8	1536.5	285.4
393216	62914560	8	1014.6	245.4
196608	125829120	16	-	572.1
393216	125829120	16	-	508.5
786432	125829120	16	-	490.7
1572864	125829120	16	-	386.3

 Tablica 2: Siatka $\sim 5GB$

A	V	n GPU	Speed[MLBUpS]	
			Rysy	Prometheus
8448	3717120	1	184.3	37.1
16896	3717121	1	150.9	36.4
Kontynuacja na następnej stronie				

Tablica 2 – kontynuacja z poprzedniej strony

A	V	n GPU	Speed[MLBUpS]	
			Rysy	Prometheus
33792	3717122	1	110.7	30.7
67584	3717123	1	68.7	22.9
16896	15728640	2	366.8	74.7
33792	15728640	2	301.0	72.5
67584	15728640	2	209.7	61.4
135168	15728640	2	136.3	45.4
33792	31457280	4	697.9	148.3
67584	31457280	4	569.9	145.6
135168	31457280	4	419.4	122.4
270336	31457280	4	284.0	90.6
540672	62914560	8	546.6	181.3
67584	62914560	8	1391.5	295.9
135168	62914560	8	1139.5	291.1
270336	62914560	8	839.2	245.3
135168	59473920	16	-	593.2
270336	59473920	16	-	579.1
540672	59473920	16	-	489.9
1081344	59473920	16	-	362.5

Tablica 3: Wymiary geometrii

n GPU	L	H	A	V	Memory[GB]
1	4096	640	12288	7864320	10.3
1	8192	320	24576	7864320	10.3
1	16384	160	49152	7864320	10.3
1	32768	80	98304	7864320	10.3
2	4096	1280	24576	15728640	10.3
Kontynuacja na następnej stronie					

Tablica 3 – kontynuacja z poprzedniej strony

n GPU	L	H	A	V	Memory[GB]
2	8192	640	49152	15728640	10.3
2	16384	320	98304	15728640	10.3
2	32768	160	196608	15728640	10.3
4	32768	320	393216	31457280	10.3
4	16384	640	196608	31457280	10.3
4	8192	1280	98304	31457280	10.3
4	4096	2560	49152	31457280	10.3
8	4096	5120	98304	62914560	10.3
8	8192	2560	196608	62914560	10.3
8	16384	1280	393216	62914560	10.3
8	32768	640	786432	62914560	10.3
16	4096	10240	196608	125829120	10.3
16	8192	5120	393216	125829120	10.3
16	16384	2560	786432	125829120	10.3
16	32768	1280	1572864	125829120	10.3
1	2816	440	8448	3717120	4.9
1	5632	220	16896	3717120	4.9
1	11264	110	33792	3717120	4.9
1	22528	55	67584	3717120	4.9
2	2816	880	16896	7434240	4.9
2	5632	440	33792	7434240	4.9
2	11264	220	67584	7434240	4.9
2	22528	110	135168	7434240	4.9
4	22528	220	270336	14868480	4.9
4	11264	440	135168	14868480	4.9
4	5632	880	67584	14868480	4.9
4	2816	1760	33792	14868480	4.9

Kontynuacja na następnej stronie

Tablica 3 – kontynuacja z poprzedniej strony

n GPU	L	H	A	V	Memory[GB]
8	2816	3520	67584	29736960	4.9
8	5632	1760	135168	29736960	4.9
8	11264	880	270336	29736960	4.9
8	22528	440	540672	29736960	4.9
16	2816	7040	135168	59473920	4.9
16	5632	3520	270336	59473920	4.9
16	11264	1760	540672	59473920	4.9
16	22528	880	1081344	59473920	4.9

Kolumny tabeli oznaczają kolejno: liczba użytych do symulacji kart GPU(**n GPU**), całkowita długość domeny obliczeniowej na osi X(**L**), całkowita długość domeny obliczeniowej na osi Y(**H**), powierzchnia przepływu informacji(**A**), całkowita objętość domeny obliczeniowej(**V**), ilość pamięci zaalokowanej na 1 karcie GPU(**Memory[GB]**)

3.1.4 Wydajność GPU

W poniższych tabelach przedstawiono dane techniczne dotyczące kart GPU, oraz ich wydajności w trakcie wykonywania symulacji na jednej karcie. Efektywność zostało obliczona wg poniższego wzoru ([Rownanie 10](#)).

$$efektywnosc = \frac{speed}{Ilosc CUDA CORE * Taktowanie} \quad (10)$$

Tablica 4: Wydajność GPU - siatka $4096 \times 640 \times 3 \sim 10GB$

	Model Karty	Taktowanie[MHz]	Cuda-Cores	MLUPs	Efektywność
Rysy	NVIDIA Tesla V100	1380	5120	193.4	2.74E-05
Prometheus	NVIDIA Tesla K40 XL	928	2880	35.9	1.34E-05

Tablica 5: Wydajność GPU - siatka $2816 \times 440 \times 3 \sim 5GB$

	Model Karty	Taktowanie[MHz]	Cuda-Cores	MLUPs	Efektywność
Rysy	NVIDIA Tesla V100	1380	5120	184.31	2.61E-05
Prometheus	NVIDIA Tesla K40 XL	928	2880	37.05	1.39E-05

3.2 Analiza wyników

Z wykresów silnego skalowania możemy zauważyc, że dla stałego rozmiaru problemu wykres załamuje się po przekroczeniu 4 kart GPU.

Liniowy rozkład punktów na wykresach słabego skalowania jest zgodny z prawem Gustafson'a.

Z wykresów [Rysunek 8](#) możemy zauważyc, że przy nieodpowiednim doborze siatki obliczeniowej mimo zwiększonej ilości procesorów działających możemy uzyskać mniejszą szybkość obliczeń. Co więcej na wykresach Speed(A/V) ([Rysunek 8](#), [Rysunek 9](#)) widać, że kształt linii nie zależy od ilości procesorów(kart GPU). Potwierdza to zależność szybkości kodu obliczeniowego opartego o metodą Lattice Boltzmann, od udziału *ghost cells* w całej objętości problemu obliczeniowego. Należy zatem zwracać uwagę na odpowiednią konfigurację siatki obliczeniowej w celu uzyskania optymalnej efektywności kodu obliczeniowego.

Bibliografia

- [1] Martin Geier, Andreas Greiner i Jan G Korvink. “A factorized central moment lattice Boltzmann method”. W: *The European Physical Journal Special Topics* 171.1 (2009), s. 55–61.
- [2] Erlend Magnus Viggen. “The lattice Boltzmann method: Fundamentals and acoustics”. W: (2014).
- [3] Martin Geier i in. “The cumulant lattice Boltzmann equation in three dimensions: Theory and validation”. W: *Computers & Mathematics with Applications* 70.4 (2015), s. 507–547.
- [4] Krüger Timm i in. “The lattice Boltzmann method: principles and practice”. W: *Springer International Publishing AG Switzerland, ISSN* (2016), s. 1868–4521.