## Теория Параллелизма

# Отчет Уравнение теплопроводности (openacc)

Выполнил 21932, Бабенко Егор Степанович

02.03.2022

#### Цель работы

Реализовать решение уравнение теплопроводности методом Якоба (пятиточечный шаблон) для двумерной сетки. Произвести профилирование программы для GPU и оптимизацию кода. Произвести сравнение времени работы на CPU и GPU.

Используемый компилятор: pgc++

Для компиляции различных версий использовались команды:

- pgc++ t2.cpp -o t2\_MultiCore.pg -fast -acc=multicore -O2 -Mconcur=allcores
- pgc++ t2.cpp -o t2\_OneCore.pg -fast -O2
- pgc++ t2.cpp -o t2\_GPU.pg -fast -acc=gpu -O2 -D OPENACC\_\_

Для дополнительной профилировки: -D NVPROF\_

Используемый профилировщики: nvprof, nsys

Nsys использовался с OpenACC trace, а также NVTX trace.

Замер времени работы всей программы производился с помощью команды time. Время высчитывалось как среднее между пятью запусками.

## CPU-onecore

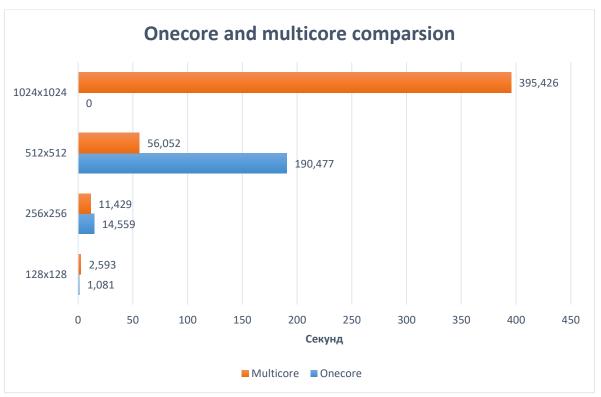
Donnan comen	Время	Ошибка	Количество
Размер сетки	выполнения (сек)	Ошиока	итераций
128x128	1.081	9.9998e-07	30074
256x256	14.559	9.9993e-07	102885
512x512	190.477	9.99984e-07	339599
1024x1024	2285.354	1.36929e-06	1000000

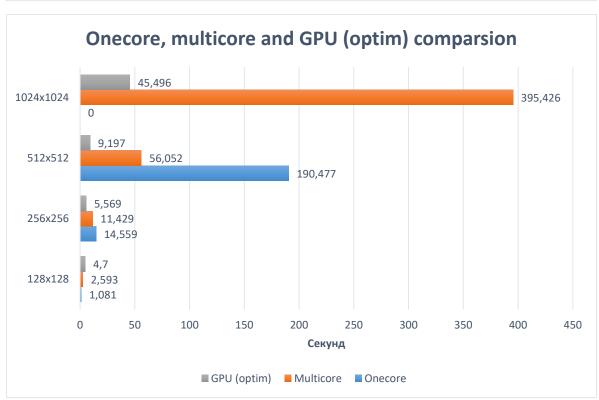
## CPU-multicore

Розмар астин	Время	Ошибка	Количество
Размер сетки	выполнения (сек)	Ошиока	итераций
128x128	2.593	9.9998e-07	30074
256x256	11.429	9.9993e-07	102885
512x512	56.052	9.99984e-07	339599
1024x1024	395.426	1.36929e-06	1000000

## **GPU-optimized**

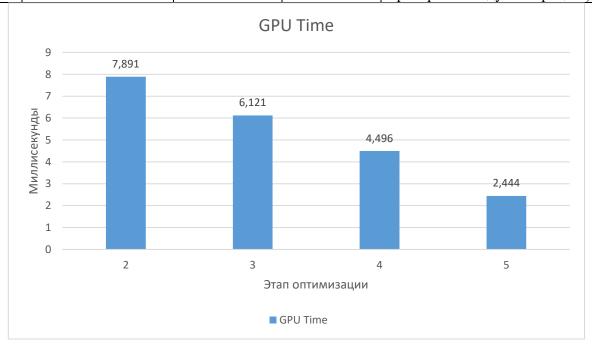
Размер сетки	Время	Ошибка	Количество
	выполнения (сек)	Ошиока	итераций
128x128	4.700	9.98757e-07	30078
256x256	5.569	9.99703e-07	102888
512x512	9.197	9.99965e-07	339600
1024x1024	45.496	1.36929e-06	1000000





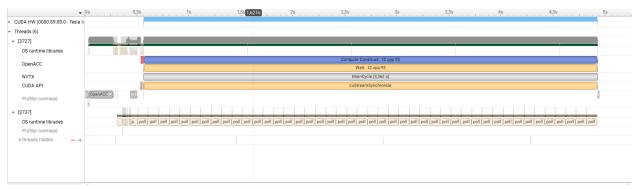
## Этапы оптимизации кода на GPU Этапы оптимизации на сетке 512x512

№	Время выполнения основного цикла	Ошибка	Кол-во итераций	Комментарий
1	4.362 c	0.107043	100	Попытка распараллелить весь цикл while (не получилось)
2	7.891 мс	0.107043	100	Распараллеливание только внутреннего цикла
3	6.121 мс	0.107043	100	Reduction на error, ручное управление памятью
4	4.496 мс	0.107043	100	Замена копирование DtoD на swap указателей внутри GPU, а также асинхронный запуск внутреннего цикла
5	2.444 мс	0.107043	100	Обновление error на хосте каждые 5 итераций (т.к. ошибка постоянно уменьшается, нет смысла проверять каждую итерацию)

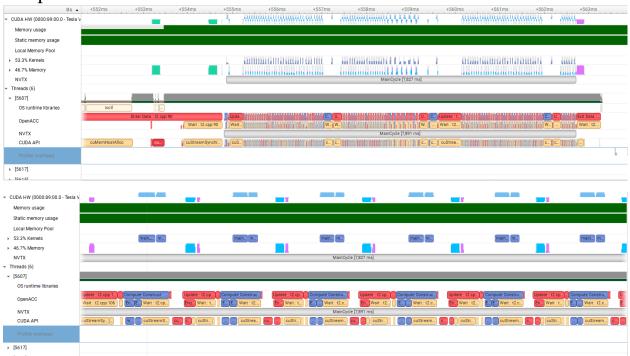


Возможной оптимизацией кода на GPU еще может быть уменьшение количества обменов данных между CPU и GPU. Т.е. делать проверку на ошибки и количества итераций на GPU, тем самым мы уменьшим количество синхронизаций до минимального количества.

### Перывый этап:



#### Второй этап:

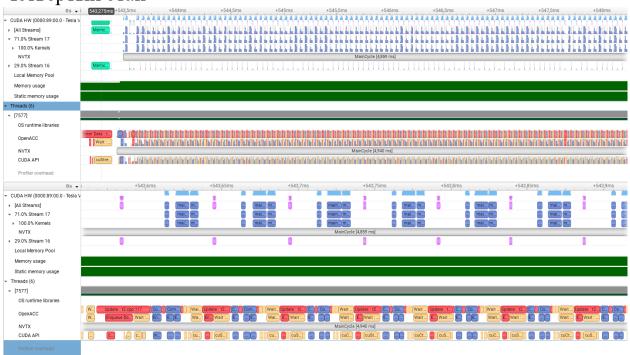


### Третий этап:

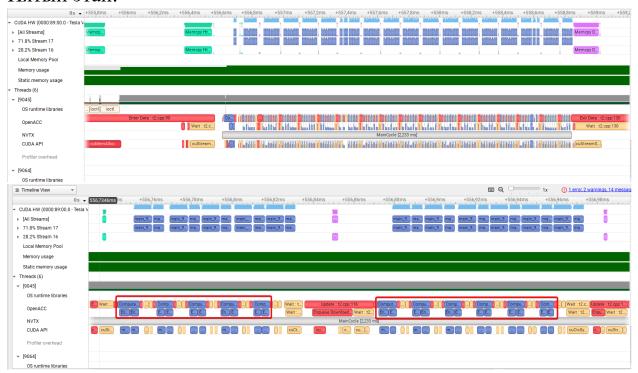




Четвертый этап



#### Пятый этап:



### Вывод:

Для обработки сетки небольшого размера будет целесообразнее использовать однопоточный вариант, т.к. издержки на инициализацию видеокарты занимают время. Но при увеличении размера сетки видно, что распараллеливание задачи более эффективно, чем выполнение в одном потоке. Т.к. среднее время выполнения одного цикла на GPU, сильно меньше, чем на CPU.

 $Github: \underline{https://github.com/JooudDoo/Parallelism-Tasks}$ 

Manufacture	143 1:	ines (120 sloc) 3.88 KB Raw Blame 🗸 🔻 🕻
Million Controls		
Michael Contents		#include (sstream)
Section of the control of the cont		
Section (cognomical manages, control of the property of the control of the company of the control of the cont		#endif
		$\label{linear_selection} \verb #include  $
intricent (committee in the content of the conten		
		void initArrays(double* mainArr, double* subArr, int& m, int& m){
		<pre>std::memset(mainArr, 0, sizeof(double)*(n)*(m));</pre>
		at(mainArr, 0, m-1) = 20;
attentor, 3,0; (contentor, 3,0; ) (contentor, 3,0) (contentor, 3,0) (contentor, 3,0);  attentor, 3,0; (contentor, 3,0; ) (contentor, 3,0) (contentor, 3,0) (contentor, 3,0);  attentor, 3,0; (contentor, 3,0; ) (contentor, 3,		
attention_pails (attention_pails).attention_pails).intention_pails).  attention_pails (attention_pails).attention_pails).intention_pails).  attention_pails (attention_pails).  buttention_pails (attention_pails).  attention_pails (attention_pails).  buttention_pails (attention_pails).  attention_pails (attention_pails).  attention_pails (attention_pails).  attention_pails (attention_pails).  attention_pails (attention_pails).  buttention_pails (attention_pails).  attention_pails (at		
Simple   State   Sta		
treintertypemen 10  treintertypemen 10  treintertypemen 10  treintertypemen 20  treint		
Transplanting common   Transplanting   Transplanting common   Transplanting common   Transplanting   Transpl		
stitutioptives strong  strong correct  result		
Treadity		std::stringstream stream;
**************************************		T result;
		<pre>throw std::invalid_argument("Wrong argument type");</pre>
constraint in THES_ETHERS_UPDATE = 5;  int main(int args, cham "exp(1))(  tool bonderstiter = elist;  double args = 15:4;  int = =0;  int = =0;  int = =0;  int = =0;  folict arg = 0; arg = args args) args    int = =0;  folict arg = 0; arg = args args    int = =0;  folict arg = 0; arg = args args    int = =0;  class   int = =0;  int = =0;  class   int = =0;  int interval = =0;  int inte		return result;
int main(int args, char "exp())(  bit interdines - files;  book seems the control of the control		constempr int ITERS_BETWEEN_UPDATE = 5;
Soci		
int iteration = 165  int = -125  int = -12		
Int = = n;		int iterations = 1E6;
<pre>implication(application(application)) implication(application(application)) implication(application(application)) implication(application(application))) implication(application(application))) implication(application(application))) implication(application(application))) implication(application)) imp</pre>		
		<pre>for(int arg = 0; arg &lt; args; args+)(    if(std::stremo(argv/acgl "-rms") == 8)f</pre>
		<pre>iterations = extractNumber<int>(argv[arg+1]);</int></pre>
# = ""  # = ""  # = ""  # = ""  # = ""  # = ""  # = ""  # = "		
std::cout << 'Character settings:" << std::end;  double* Frew = new double(n*m);  double* Frew = new double(n*m);  double* Frew = new double(n*m);  double error = 0;  int iteration = 0;  int iteration = 0;  int iteration = 0;  int iteration = 0;  frequen acc enter data copyin(fined;:n*m), F[:n*m], error)  std::end MORDE.  mcusingePush("Mointycle*);  endif  do {		showResultArr = true;
<pre>std:cout &lt;&lt; Current settings: &lt;&lt; std:read; std:cout &lt;&lt; Cuttive: &lt;= ceps &lt;&lt; distribution; std:cout &lt;&lt; Cuttive: iteration: </pre> std:cout << Cuttive: iteration:  std:cout << Cuttive: <pre>cep </pre> std:cout  std: std:cout  std:cout <pre>std:cout &lt;= cout &lt;</pre>		
<pre>std::cost &lt;&lt; "ttbs: iteration: &lt;&lt; ttreat iteration: &lt;&lt; ttreat iteration: </pre> std::cost << "ttbs: <pre>content </pre> std::cost << 'ttbs: <pre>content </pre> std::cost < std::cost  std::cost <pre>cost <pre>cost </pre> std::cost <pre>cost <pre>cost </pre> std::cost <pre>cost <pre>cost</pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>		
double* finew = new double[ofws];  double* finew = new double[ofws];  inithropys(F, Fnew, n, m);  double error = 8;  int iteration = 6;  int iteration = iteration = 6;  int iteration = 6		<pre>std::cout &lt;&lt; "\tMax iteration: " &lt;&lt; iterations &lt;&lt; std::endl;</pre>
<pre>double* Free = new Souble(n*m);  intitrreys(f, free, n, m);  double error = 0;  int iteration = 0;  int iteration = 0;  int iteration = 0;  int itersfetweenipdate = 0;  #pragum acc enter data copyin(free[in*m], F[:n*m], error)  #rider IMPROF_</pre>		
<pre>double error = 0; int iteration = 0;  int</pre>		<pre>double* Fnew = new double[n*m];</pre>
<pre>int iteration = 0;  int itersfetweenipdate = 0;  sprague acc enter data copyin(frew[in*m], F[:n*m], error)  #irder MPROF_ mytsAmgePusk("MainCycle");  #endif  do {</pre>		
## ## ## ## ## ## ## ## ## ## ## ## ##		
### ### ##############################		<pre>int itersBetweenUpdate = 0;</pre>
# dider MPMOF_		#pragma acc enter data copyin(Fnew[:n*m], F[:n*m], error)
Secolif		
# progna acc parallel present(error) esync(0)  (		#endif do {
### Spragma acc parallel loop collapse(2) present(fnew[:n*m], F[:n*m], error) reduction(max:error)  ### ### ### ### ### ### ### ### ### #		<pre>#pragma acc parallel present(error) async(0) {</pre>
# # # # # # # # # # # # # # # # # # #		
<pre>for(int) y= 1; y &lt; ==1; y+&gt;{</pre>		
		for(int y= 1; y < m-1; y++){
106		error = fmax(error, fabs(at(Fnew,x,y) - at(F,x,y)));
F = Free;		
		F = Fnew;
<pre>111</pre>		
<pre>in dendif if(iterstetween)pdate &gt;= ITERS_BETMENLEPOATE &amp;&amp; iteration &lt; iterations)(</pre>		<pre>acc_attach((void**)F);</pre>
		#endif
117		#pragma acc update self(error) wait
120		
### ### ##############################		itersBetweenUpdate++;
### ### ##############################		#ifdef INVPROF_
### ### ##############################		
130		#pragma acc exit data delete(fnew[:n*m]) copyout(f[:n*m], error)
132		std::cout << "Error: " << error << std::endl;
135 } 136		<pre>for(int x = 0; x &lt; n &amp;&amp; showResultArr; x++){     for(int y = 0; y &lt; m; y++){</pre>
137 ) 130 delete[] F; 140 delete[] Fnew; 141 turn 0;		
139 delete[] F; 140 delete[] Fnew; 141 142 return 0;		