

МГУ имени М. В. Ломоносова
Механико-математический факультет

Презентация к курсовой работе

Реализация библиотеки параллельной записи больших
файлов с вещественными числами в текстовом
представлении

Нагорных Яна

Москва – 2018

Печать больших массивов чисел без округления с большой точностью всегда занимает много времени. Однако, не вся печать упирается в возможности диска, как это может показаться. Кроме того, у печати данных мало ресурсов для ускорения.

Цели работы:

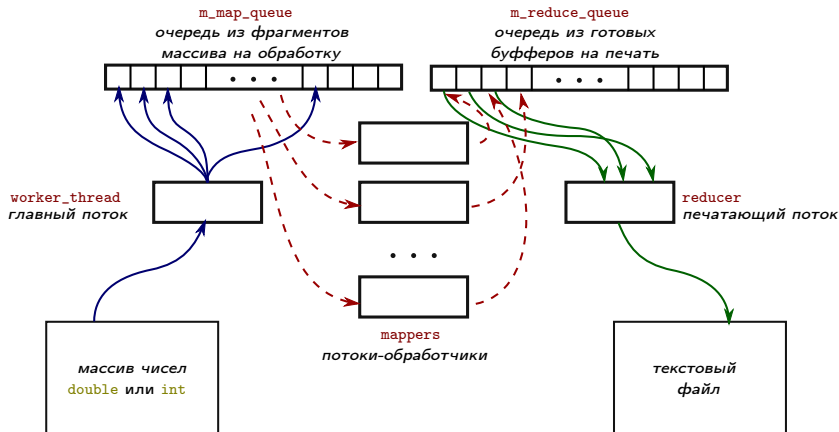
1. Ускорить печать больших массивов без потери точности;
2. Использовать быстрые алгоритмы печати целых чисел и чисел с плавающей точкой.

Возможные варианты улучшений:

- Применение более быстрых алгоритмов преобразования чисел в строки
- Использование многопоточного программирования
- Изменение формата вывода (отбрасывание лишних нулей, сокращенная запись повторяющихся чисел)

Распределение задач

Наглядно работа потоков изображена на рисунке:



Алгоритм Grisu

- Выражаем v :

$$v = \frac{f_v}{2^{-e_v}}.$$

- Десятичные цифры v могут быть вычислены путем нахождения десятичного показателя t , для которого

$$1 \leq \frac{f_v \times 10^t}{2^{-e_v}} < 10$$

- Идея Grisu состоит в кэшировании приблизительных значений дробей $\frac{10^t}{2^{e_t}}$.

Алгоритм Grisu2

- У Grisu есть существенный недостаток. Например, при значениях по умолчанию число 1 будет напечатано в виде $10000000000000000000e-19$.
- В отличие от Grisu алгоритм Grisu2 не генерирует полное десятичное представление, а просто возвращает значащие цифры и соответствующий показатель. Затем процедура форматирования объединяет эти данные для получения представления в требуемом формате.

- Grisu2 использует дополнительные флаги для создания более короткой выходной строки.
- Также Grisu2 не будет работать с точными числами, а вместо этого будет вычислять аппроксимации m^- и m^+ – *ближайшие числа в памяти*.
- Во избежание ошибочных результатов, увеличивается диапазон, в котором, согласно алгоритму, может оказаться полученное число.

```
array[0] = 1;  
array[1] = 1.2;  
array[2] = 1.23;  
array[3] = 1.23400000;  
array[4] = 1.23456789;  
array[5] = -1;  
array[6] = -1.234;  
array[7] = sqrt (2);  
array[8] = 1234e-36;  
array[9] = 0.000000123;  
array[10] = 0.123;  
array[11] = 12.3;  
array[12] = 123.000;
```

МАССИВ



```
1  
1.2  
1.23  
1.234  
1.23456789  
-1  
-1.234  
1.4142135623730952  
1.234e-33  
1.23e-7  
0.123  
12.3  
123
```

ВЫХОДНОЙ файл

Улучшения

- Если в массиве есть n подряд идущих одинаковых чисел с заданной точностью, то есть $\forall i : 1 \leq i < n$ верно, что $\|x_i - x_{i-1}\| \leq \varepsilon$, то сократим запись n чисел и вернем строку вида $n*x$.
- Запись целых чисел, означающих количество повторяющихся элементов массива, также можно ускорить. Суть алгоритма заключается в быстром логарифмировании числа по основанию 10.

Случайные числа

Размер массива	Число потоков				Стандартная печать	Размер файла
	6	4	2	1		
10^7	0.593	0.880	1.620	3.196	4.256	245 MB
	0.562	0.841	1.612	3.239	4.176	
	0.530	0.802	1.502	3.052	4.188	
$5 \cdot 10^7$	2.571	4.044	7.812	15.37	22.47	1.2 GB
	2.634	4.273	8.214	16.30	21.11	
	2.689	4.179	7.822	15.32	20.89	
10^8	5.219	8.276	15.67	32.46	41.71	2.4 GB
	5.077	7.970	15.30	30.57	41.78	
	5.189	8.078	15.37	30.75	41.96	
$5 \cdot 10^8$	41.12	49.15	75.23	148.91	200.94	12 GB
	40.23	50.08	75.92	149.08	200.33	
	41.23	49.16	74.92	148.52	200.69	

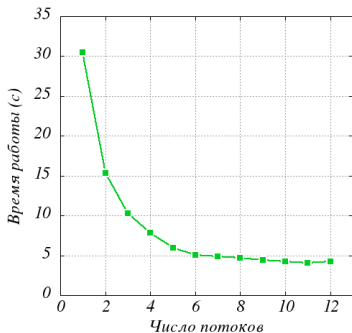
Среднее ускорение работы алгоритма:

Размер массива	Число потоков			
	6	4	2	1
10^7	7.49	5.00	2.67	1.33
$5 \cdot 10^7$	8.17	5.16	2.70	1.37
10^8	8.10	5.16	2.71	1.34
$5 \cdot 10^8$	4.91	4.06	2.66	1.35

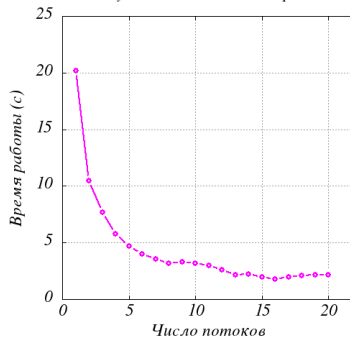
Ускорение на одном потоке демонстрирует ускорение работы Grisu2 по сравнению со стандартной печатью.

Наглядно зависимость времени от числа потоков для массива размером 10^8 изображена на графиках.

Запуск на машине с 6 ядрами



Запуск на машине с 20 ядрами



Целые числа

Размер массива	Число потоков				Станд. печать	Размер файла
	6	4	2	1		
10^7	0.213	0.322	0.643	1.297	5.300	56 MB / 205 MB
	0.216	0.320	0.649	1.284	5.245	
	0.219	0.337	0.650	1.321	5.312	
$5 \cdot 10^7$	1.052	1.664	3.319	6.362	27.93	295 MB / 1 GB
	1.069	1.675	3.334	6.375	29.46	
	1.071	1.662	3.340	6.490	29.43	
10^8	2.057	3.374	6.618	12.61	55.04	590 MB / 2 GB
	2.018	3.309	6.712	12.63	55.70	
	2.012	3.248	6.601	13.66	56.31	
$5 \cdot 10^8$	10.82	16.68	32.14	62.94	283.61	2.9 GB / 10 GB
	10.91	16.68	32.20	64.08	290.70	
	10.15	16.45	32.09	64.12	287.54	

Размер файла, полученного с помощью нового алгоритма гораздо меньше размера файла, полученного стандартной печатью, так как отброшены лишние нули. За счет этого ускорение возросло:

Размер массива	Число потоков			
	6	4	2	1
10^7	24.47	16.19	8.17	4.06
$5 \cdot 10^7$	27.20	17.36	8.69	4.51
10^8	27.44	16.82	8.38	4.29
$5 \cdot 10^8$	27.03	17.30	8.93	4.51

Повторяющиеся числа

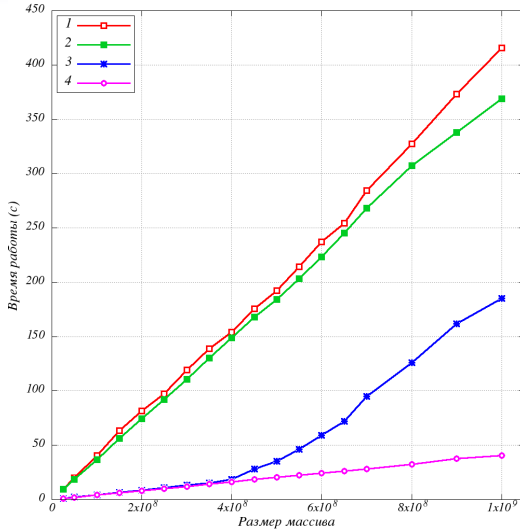
Размер массива	Число потоков				Станд. печать	Размер файла
	6	4	2	1		
10^7	0.112	0.181	0.339	0.652	3.445	24 MB / 187 MB
	0.109	0.159	0.310	0.604	3.322	
	0.119	0.168	0.329	0.661	3.460	
$5 \cdot 10^7$	0.521	0.843	1.630	2.983	16.91	123 MB / 936 MB
	0.549	0.841	1.643	3.067	17.30	
	0.530	0.833	1.612	2.875	16.96	
10^8	1.178	1.748	3.343	6.056	36.19	245 MB / 1.8 GB
	1.152	1.678	3.209	5.959	36.38	
	1.163	1.689	3.290	6.039	36.41	
$5 \cdot 10^8$	5.720	8.354	15.82	31.50	182.22	1.2 GB / 9.4 GB
	5.714	8.346	15.71	31.29	182.00	
	5.816	8.418	15.92	31.69	183.52	

За счет того, что все последовательности одинаковых подряд идущих чисел будут сворачиваться в короткую строку вида $n \cdot x$, уменьшился файл и увеличилось ускорение.

Размер массива	Число потоков			
	6	4	2	1
10^7	30.08	20.13	10.46	5.33
$5 \cdot 10^7$	31.98	20.32	10.47	5.74
10^8	31.20	21.31	11.07	6.03
$5 \cdot 10^8$	31.75	21.81	11.54	5.80

Огромные массивы случайных чисел





- Сравним стандартную печать и алгоритм, запущенный на 12 (+2) потоках.
- Помимо обычного запуска, проведем и запуск с записью не на диск, а в разделяемую память *shared-memory*.
- На следующем графике приведена зависимость времени работы от размера массива.



1 – стандартная печать с записью на диск; 2 – стандартная печать с записью в разделяемую память; 3 – алгоритм параллельной печати с записью на диск; 4 – алгоритм параллельной печати с записью в разделяемую память.

- В результате написания курсовой работы была решена поставленная задача: реализована библиотека параллельной записи массивов вещественных чисел.
- В ходе тестирования была проверена точность работы реализованного алгоритма, а также измерено ускорение в сравнении со стандартной функцией печати.
- Написанная на языке C++ подпрограмма была внедрена в промышленный гидродинамический симулятор tNavigator.

Список использованной литературы

-  FLORIAN LOITSCH. Printing Floating-Point Numbers Quickly and Accurately with Integers, 2004.
-  WOJCIECH MUŁA. SSE: conversion integers to decimal representation, 2011.
-  БОГАЧЕВ К.Ю.. Основы параллельного программирования. – М.: Бинوم. Лаборатория знаний, 2010.
-  DAVID GOLDBERG. What every computer scientist should know about floating-point arithmetic. – ACM Computing Surveys, 23(1): 5–48, 1991.

Спасибо за внимание!