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

Нагорных Яна Валерьевна
студент 410 группы

Научный руководитель — д. ф.-м. н., доцент К.Ю. Богачев

Москва, 2018г.

Введение

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)

Описание алгоритма

Распределение задач

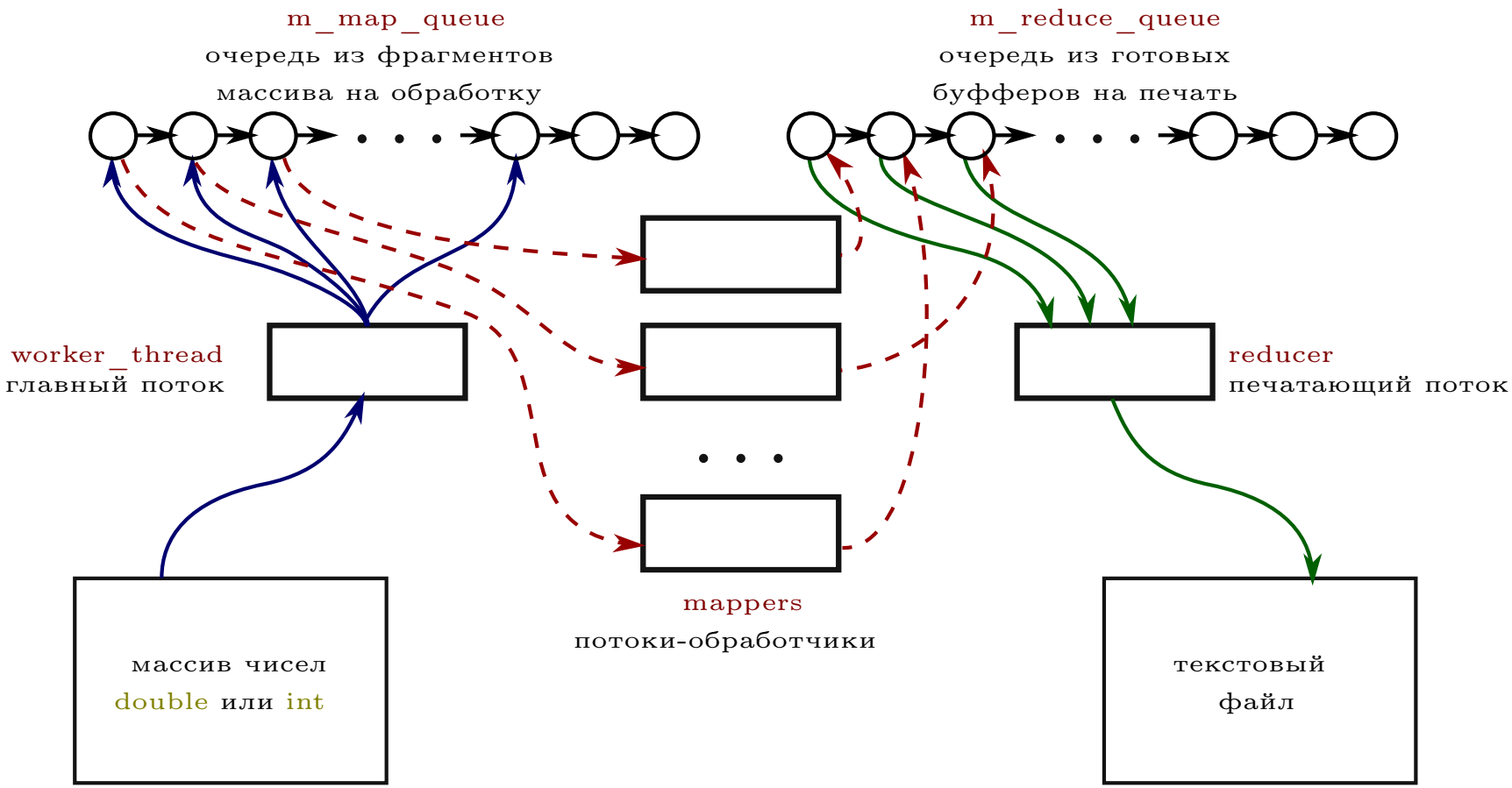


Рисунок 1. Работа потоков

Алгоритм Grisu

- Выражаем v:

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

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

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

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

Алгоритм Grisu2

- У Grisu есть существенный недостаток: все числа выводятся в экспонециальном виде.
- 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.

Пример работы Grisu2

массив		выходной файл
array[0] = 1;		1
array[1] = 1.2;		1.2
array[2] = 1.23;		1.23
array[3] = 1.23400000;		1.234
array[4] = 1.23456789;		1.23456789
array[5] = -1;		-1
array[6] = -1.234;	$\longrightarrow$	-1.234
array[7] = sqrt (2);		1.414213562373095
array[8] = 1234e-36;		1.234e-33
array[9] = 0.000000123;		1.23e-7
array[10] = 0.123;		0.123
array[11] = 12.3;		12.3
array[12] = 123.000;		123
array[13] = 1.234e2;		12.34

Рисунок 2. Полученный с помощью Grisu2, выходной файл для данного массива.

Результаты работы

Случайные числа. Среднее время работы в секундах:

Размер массива	Число потоков				Стандартная печать	Размер файла
	6	4	2	1		
10^7	0.562	0.841	1.578	3.162	4.207	245 MB
$5 \cdot 10^7$	2.667	4.223	8.070	15.91	21.79	1.2 GB
10^8	5.167	8.112	15.45	31.24	41.86	2.4 GB
$5 \cdot 10^8$	40.80	49.35	75.31	148.39	200.33	12 GB

Среднее ускорение работы алгоритма приведено в таблице далее. Ускорение на одном потоке демонстрирует ускорение работы Grisu2 по сравнению со стандартной печатью.

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

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

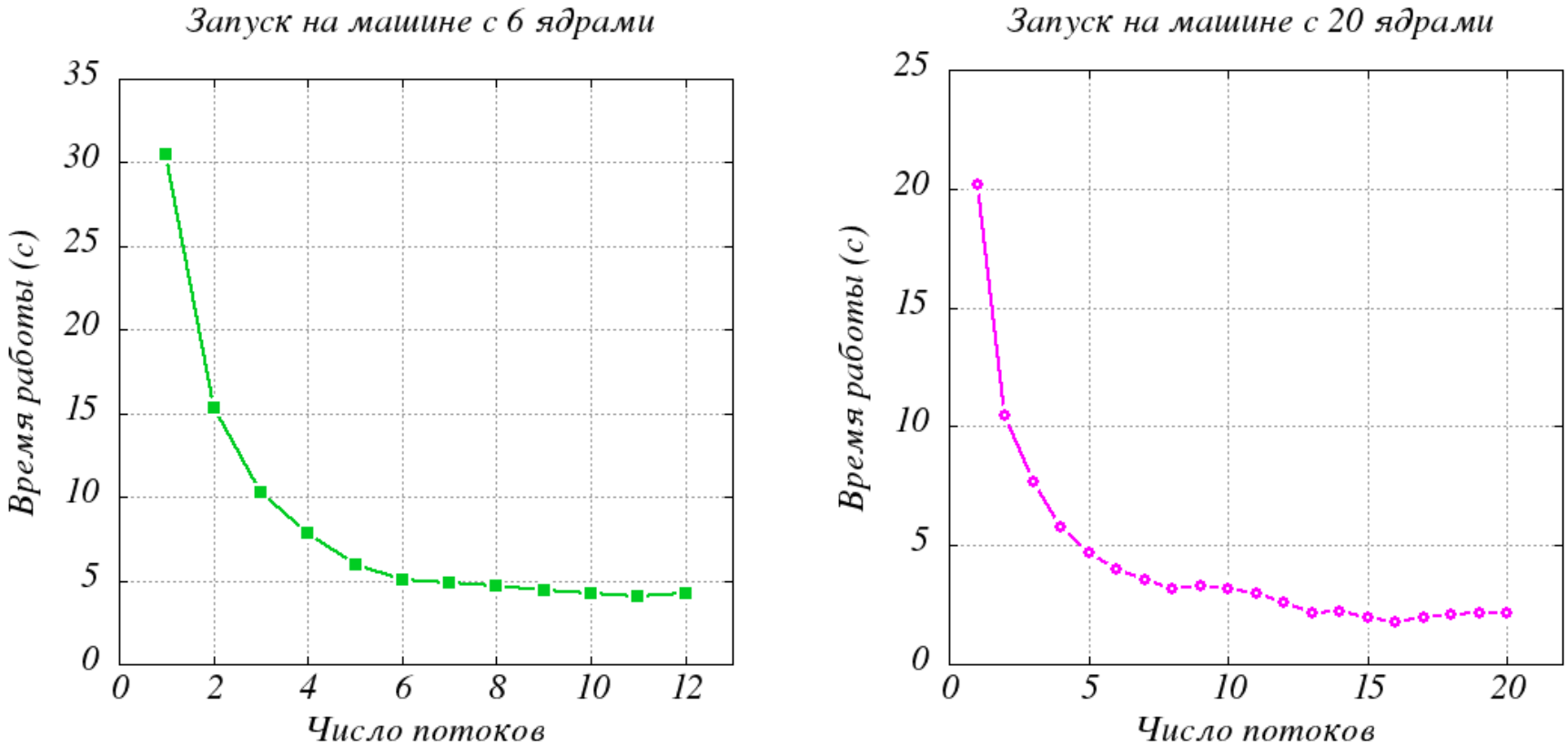


Рисунок 3. Зависимость времени работы от числа потоков.

Целые числа

Размер массива	Число потоков				Станд. печать	Размер файла
	6	4	2	1		
10^7	0.237	0.352	0.695	1.400	5.686	56 MB / 205 MB
$5 \cdot 10^7$	1.112	1.674	3.344	6.443	29.06	295 MB / 1 GB
10^8	2.106	3.330	6.628	12.95	55.54	590 MB / 2 GB
$5 \cdot 10^8$	10.86	16.57	32.11	63.57	286.70	2.9 GB / 10 GB

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	23.98	16.19	8.17	4.06
$5 \cdot 10^7$	26.15	17.36	8.69	4.51
10^8	26.37	16.82	8.38	4.29
$5 \cdot 10^8$	26.38	17.30	8.93	4.51

Повторяющиеся числа

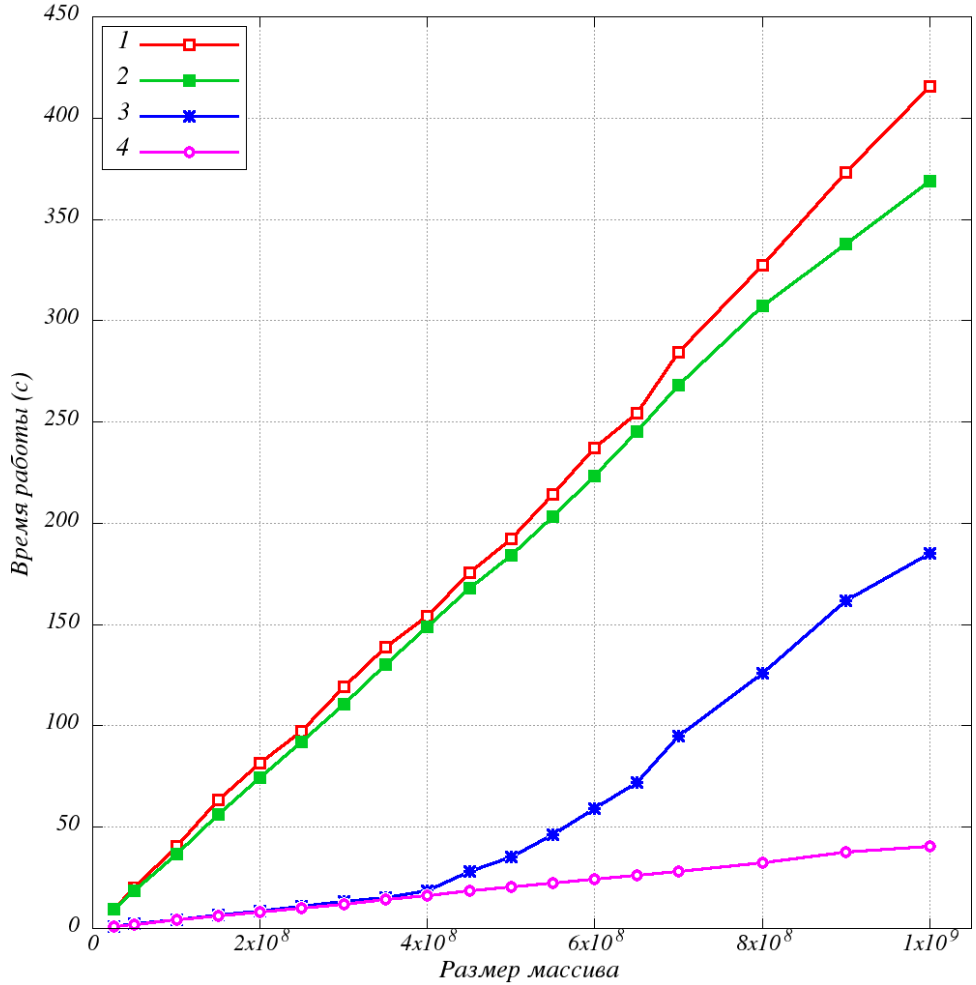
Размер массива	Число потоков				Станд. печать	Размер файла
	6	4	2	1		
10^7	0.114	0.169	0.328	0.643	3.422	24 MB / 187 MB
$5 \cdot 10^7$	0.535	0.840	1.633	2.97	17.10	123 MB / 936 MB
10^8	1.164	1.704	3.280	6.022	36.31	245 MB / 1.8 GB
$5 \cdot 10^8$	5.748	8.368	16.23	31.46	182.50	1.2 GB / 9.4 GB

За счет того, что все последовательности одинаковых подряд идущих чисел будут сворачиваться в короткую строку вида n*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.

Заключение

- 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.

Спасибо за внимание!

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.: Бином. Лаборатория знаний, 2010.
- David Goldberg. What every computer scientist should know about floating-point arithmetic. – ACM Computing Surveys, 23(1): 5–48, 1991.