Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Сибирский Государственный Университет Телекоммуникаций и Информатики СибГУТИ

Кафедра Вычислительных систем

Лабораторная работа №3
По дисциплине “Архитектура вычислительных систем”

Выполнил:
Студент группы ИВ-921
Ярошев Р. А..

Работу проверил:
Ассистент кафедры ВС
Петухова Я.В.

Новосибирск 2021

Задание

Разработать программу (benchmark) для оценки производительности подсистемы памяти.

1. Написать программу(функцию) на языке C/C++/C# для оценки производительности подсистемы памяти.

На вход программы подать следующие аргументы.
1) Подсистема памяти. Предусмотреть возможность указать подсистему для проверки
производительности: RAM (оперативная память), HDD/SSD и flash.
2) Размер блока данных в байтах, Кб или Мб. Если размерность не указана, то в байтах, если указана,
то соответственно в Кбайтах или Мбайтах.
3) Число испытаний, т.е. число раз повторений измерений.
Пример вызова программы: ./memory_test –m RAM –b 1024|1Kb –l 10 или
./memory_bandwidth —memory-type RAM|HDD|SSD|flash
—block-size 1024|1Kb
—launch-count 10
В качестве блока данных использовать одномерный массив, в котором произведение числа элементов
на их размерность равна требуемому размеру блока данных. Массив инициализировать случайными
значениями. Для тестирования HDD/SSD и flash создать в программе файлы в соответствующих
директориях.
Измерение времени реализовать с помощью функции clock_gettime() или аналогичной с точность до
наносекунд. Измерять время исключительно на запись элемента в память или считывание из неё, без
операций генерации или преобразования данных.

На выходе программы в одну строку CSV файла со следующей структурой:
[MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWriteTime;WriteBandwidth;
AbsError(write);RelError(write);ReadTime;AverageReadTime;ReadBandwidthAb

sError(read);RelError(read);], где
MemoryType – тип памяти (RAM|HDD|SSD|flash) или модель устройства, на котором проводятся испытания;
BlockSize – размер блока данных для записи и чтения на каждом испытании;
ElementType – тип элементов используемых для заполнения массива данных;
BufferSize – размер буфера, т.е. порции данных для выполнения одно операции записи или чтения;
LaunchNum – порядковый номер испытания;
Timer – название функции обращения к таймеру (для измерения времени);
WriteTime – время выполнения отдельного испытания с номером LaunchNum [секунды];
AverageWriteTime – среднее время записи из LaunchNum испытаний [секунды];
WriteBandwidth – пропускная способность памяти (BLOCK_SIZE/AverageWriteTime) * 106 [Mb/s]
AbsError(write) – абсолютная погрешность измерения времени записи или СКО [секунды];
RelError(write) – относительная погрешность измерения времени [%];
ReadTime – время выполнения отдельного испытания LaunchNum [секунды];
AverageReadTime – среднее время записи из LaunchNum испытаний [секунды];
ReadBandwidth – пропускная способность памяти (BLOCK_SIZE/AverageReadTime) * 106 [Мб/сек.]
AbsError(read) – абсолютная погрешность измерения времени чтения или СКО [секунды];
RelError(read) – относительная погрешность измерения времени [%].

2. Написать программу(функцию) на языке C/C++/C# или скрипт (benchmark) реализующий серию
испытаний программы(функции) из п.1. Оценить пропускную способность оперативной памяти при
работе с блоками данных равными объёму кэш-линии, кэш-памяти L1, L2 и L3 уровня и превышающего
его. Для HDD|SSD и flash провести серию из 20 испытаний с блоками данных начиная с 4 Мб с шагом
4Мб. Результаты всех испытаний сохранить в один CSV файл со структурой, описанной в п.1.

* Для HDD|SSD и flash оценить влияние размера буфера (BufferSize) на пропускную способность памяти.

Курс «Архитектура вычислительных систем». СибГУТИ. 2020 г.
3. На основе CSV файла построить сводные таблицы и диаграммы отражающие:
1) Зависимость пропускной способности записи и чтения от размера блока данных (BlockSize) для
разного типа памяти;
2) Зависимость погрешности измерения пропускной способности от размера блока данных для
разного типа памяти;
3) Зависимость погрешности измерений от числа испытаний LaunchNum;
4) * Зависимость пропускной способности памяти от размера буфера для HDD|SSD и flash памяти;
4. ** Оценить пропускную способность файла подкачки (windows) или раздела SWAP (linux). Сравнить с
пропускной способностью RAM, HDD/SSD и flash.

Результаты работы

Проведена серия испытаний для оценки пропускной способности оперативной памяти. Оценивались блоки данных размером кэш-линии, L1, L2 и L3 (уровни кеша), а также превышающим уровни кеша.

MemoryType	BlockSize	ElementType	BufferSize	LaunchNum	Timer	WriteTime	AverageWriteTime	WriteBandwidth	AbsError(write)	RelError(write)	ReadTime	AverageReadTime	ReadBandwidth	AbsError(read)	RelError(read)
RAM	64	uint8_t	1	1	clock()	1.000000e-06	6.666667e-07	9.600000e+13	4.714045e-07	3.333333e-05%	1.000000e-06	6.666667e-07	9.600000e+13	4.714045e-07	3.333333e-05%
RAM	64	uint8_t	1	2	clock()	1.000000e-06	6.666667e-07	9.600000e+13	4.714045e-07	3.333333e-05%	1.000000e-06	6.666667e-07	9.600000e+13	4.714045e-07	3.333333e-05%
RAM	64	uint8_t	1	3	clock()	0.000000e+00	6.666667e-07	9.600000e+13	4.714045e-07	3.333333e-05%	0.000000e+00	6.666667e-07	9.600000e+13	4.714045e-07	3.333333e-05%
RAM	32768	uint8_t	1	1	clock()	5.300000e-05	4.700000e-05	6.971915e+14	4.242641e-06	3.829787e-05%	5.300000e-05	4.700000e-05	6.971915e+14	4.242641e-06	3.829787e-05%
RAM	32768	uint8_t	1	2	clock()	4.400000e-05	4.700000e-05	6.971915e+14	4.242641e-06	3.829787e-05%	4.400000e-05	4.700000e-05	6.971915e+14	4.242641e-06	3.829787e-05%
RAM	32768	uint8_t	1	3	clock()	4.400000e-05	4.700000e-05	6.971915e+14	4.242641e-06	3.829787e-05%	4.400000e-05	4.700000e-05	6.971915e+14	4.242641e-06	3.829787e-05%
RAM	262144	uint8_t	1	1	clock()	6.670000e-04	5.273333e-04	4.971125e+14	1.049518e-04	2.088791e-03%	6.670000e-04	5.273333e-04	4.971125e+14	1.049518e-04	2.088791e-03%
RAM	262144	uint8_t	1	2	clock()	4.140000e-04	5.273333e-04	4.971125e+14	1.049518e-04	2.088791e-03%	4.140000e-04	5.273333e-04	4.971125e+14	1.049518e-04	2.088791e-03%
RAM	262144	uint8_t	1	3	clock()	5.010000e-04	5.273333e-04	4.971125e+14	1.049518e-04	2.088791e-03%	5.010000e-04	5.273333e-04	4.971125e+14	1.049518e-04	2.088791e-03%
RAM	12582912	uint8_t	1	1	clock()	2.172800e-02	2.114433e-02	5.950962e+14	4.292477e-04	8.714087e-04%	2.172800e-02	2.114433e-02	5.950962e+14	4.292477e-04	8.714087e-04%
RAM	12582912	uint8_t	1	2	clock()	2.099700e-02	2.114433e-02	5.950962e+14	4.292477e-04	8.714087e-04%	2.099700e-02	2.114433e-02	5.950962e+14	4.292477e-04	8.714087e-04%
RAM	12582912	uint8_t	1	3	clock()	2.070800e-02	2.114433e-02	5.950962e+14	4.292477e-04	8.714087e-04%	2.070800e-02	2.114433e-02	5.950962e+14	4.292477e-04	8.714087e-04%
RAM	12845056	uint8_t	1	1	clock()	2.174500e-02	2.126133e-02	6.041510e+14	3.574972e-04	6.011110e-04%	2.174500e-02	2.126133e-02	6.041510e+14	3.574972e-04	6.011110e-04%
RAM	12845056	uint8_t	1	2	clock()	2.089200e-02	2.126133e-02	6.041510e+14	3.574972e-04	6.011110e-04%	2.089200e-02	2.126133e-02	6.041510e+14	3.574972e-04	6.011110e-04%
RAM	12845056	uint8_t	1	3	clock()	2.114700e-02	2.126133e-02	6.041510e+14	3.574972e-04	6.011110e-04%	2.114700e-02	2.126133e-02	6.041510e+14	3.574972e-04	6.011110e-04%

Таблица. 1. Оценка кеша

Как видно из Табл. 1., время выполнения (WriteTime) отдельного испытания не превышает 5.3 секунды (при обработке 32 768 бит)

Пропускная способность оперативной памяти при чтении/записи (Read/Write) блока размером 64 бит — 9.6 Mb/s.

Пропускная способность оперативной памяти при чтении/записи (Read/Write) блока размером 32768 бит — 6.97 Mb/s .

Пропускная способность оперативной памяти при чтении/записи (Read/Write) блока размером 262144 бит — 6.27 Mb/s.

Пропускная способность оперативной памяти при чтении/записи (Read/Write) блока размером 12582912 бит — 6.21 Mb/s.

Пропускная способность оперативной памяти при чтении/записи (Read/Write) блока размером 12845056 бит — 6.26 Mb/s.

Затем была проведена аналогичная серия из 20 испытаний для SSD и flash соответсвенно. Блоки данных от 4 Мб с шагом 4 Мб.

MemoryType	BlockSize	ElementType	BufferSize	LaunchNum	Timer	WriteTime	AverageWriteTime	WriteBandwidth	AbsError(write)	RelError(write)	ReadTime	AverageReadTime	ReadBandwidth	AbsError(read)	RelError(read)
SSD	4194304	uint8_t	4194304	1	clock()	1.602000e-03	1.602000e-03	2.618167e+15	1.436346e-07	1.6467464e-07%	3.170000e-04	3.170000e-04	1.323124e+16	1.436346e-07	1.6467464e-07%
SSD	8388608	uint8_t	8388608	1	clock()	3.250000e-03	3.250000e-03	2.581110e+15	1.465346e-07	2.0446464e-07%	8.530000e-04	8.530000e-04	9.834242e+15	1.465346e-07	2.0446464e-07%
SSD	12582912	uint8_t	12582912	1	clock()	4.887000e-03	4.887000e-03	2.574772e+15	1.678646e-07	1.6467464e-07%	1.523000e-03	1.523000e-03	8.261925e+15	1.678646e-07	1.6467464e-07%
SSD	16777216	uint8_t	16777216	1	clock()	6.536000e-03	6.536000e-03	2.566894e+15	1.787895e-07	1.9787464e-07%	2.234000e-03	2.234000e-03	7.509944e+15	1.787895e-07	1.9787464e-07%
SSD	20971520	uint8_t	20971520	1	clock()	8.220000e-03	8.220000e-03	2.551280e+15	2.657447e-07	0.6467464e-07%	2.778000e-03	2.778000e-03	7.549143e+15	2.657447e-07	0.6467464e-07%
SSD	25165824	uint8_t	25165824	1	clock()	1.010500e-02	1.010500e-02	2.490433e+15	2.976857e-07	0.3532464e-07%	3.421000e-03	3.421000e-03	7.356277e+15	2.976857e-07	0.3532464e-07%
SSD	29360128	uint8_t	29360128	1	clock()	1.137100e-02	1.137100e-02	2.582018e+15	1.946744e-07	0.1259464e-07%	4.023000e-03	4.023000e-03	7.298068e+15	1.946744e-07	0.1259464e-07%
SSD	33554432	uint8_t	33554432	1	clock()	1.322500e-02	1.322500e-02	2.537197e+15	1.433535e-07	0.6536463e-07%	4.645000e-03	4.645000e-03	7.223774e+15	1.433535e-07	0.6536463e-07%
SSD	37748736	uint8_t	37748736	1	clock()	1.466600e-02	1.466600e-02	2.573894e+15	1.963636e-07	1.6536463e-07%	5.108000e-03	5.108000e-03	7.390121e+15	1.963636e-07	1.6536463e-07%
SSD	41943040	uint8_t	41943040	1	clock()	1.624700e-02	1.624700e-02	2.581587e+15	3.857855e-07	1.3985463e-07%	5.720000e-03	5.720000e-03	7.332699e+15	3.857855e-07	1.3985463e-07%
SSD	46137344	uint8_t	46137344	1	clock()	1.809300e-02	1.809300e-02	2.550011e+15	4.896855e-07	1.6536463e-07%	7.552000e-03	7.552000e-03	6.109288e+15	4.896855e-07	1.6536463e-07%
SSD	50331648	uint8_t	50331648	1	clock()	1.965100e-02	1.965100e-02	2.561277e+15	3.967858e-07	2.9842984e-07%	1.725900e-02	1.725900e-02	2.916255e+15	3.967858e-07	2.9842984e-07%
SSD	54525952	uint8_t	54525952	1	clock()	2.117000e-02	2.117000e-02	2.575624e+15	2.746363e-07	3.5863636e-07%	1.751300e-02	1.751300e-02	3.113456e+15	2.746363e-07	3.5863636e-07%
SSD	58720256	uint8_t	58720256	1	clock()	2.278500e-02	2.278500e-02	2.577145e+15	1.857474e-07	3.6536463e-07%	8.152000e-03	8.152000e-03	7.203172e+15	1.857474e-07	3.6536463e-07%
SSD	62914560	uint8_t	62914560	1	clock()	2.458400e-02	2.458400e-02	2.559167e+15	1.423587e-07	4.6455636e-07%	2.176700e-02	2.176700e-02	2.890364e+15	1.423587e-07	4.6455636e-07%
SSD	67108864	uint8_t	67108864	1	clock()	2.606300e-02	2.606300e-02	2.574871e+15	2.756856e-07	3.4372842e-07%	2.331400e-02	2.331400e-02	2.878479e+15	2.756856e-07	3.4372842e-07%
SSD	71303168	uint8_t	71303168	1	clock()	2.780600e-02	2.780600e-02	2.564309e+15	2.785785e-07	3.2423525e-07%	2.489800e-02	2.489800e-02	2.863811e+15	2.785785e-07	3.2423525e-07%
SSD	75497472	uint8_t	75497472	1	clock()	2.929200e-02	2.929200e-02	2.577409e+15	1.249636e-07	2.6356463e-07%	9.877000e-03	9.877000e-03	7.643766e+15	1.249636e-07	2.6356463e-07%
SSD	79691776	uint8_t	79691776	1	clock()	3.083300e-02	3.083300e-02	2.584626e+15	2.645789e-07	3.6486944e-07%	1.080400e-02	1.080400e-02	7.376136e+15	2.645789e-07	3.6486944e-07%
SSD	83886080	uint8_t	83886080	1	clock()	3.279300e-02	3.279300e-02	2.558048e+15	2.253463e-07	4.9876563e-07%	1.160900e-02	1.160900e-02	7.225952e+15	2.253463e-07	4.9876563e-07%
flash	4194304	uint8_t	4194304	1	clock()	3.144000e-03	3.144000e-03	1.334066e+15	0.536346e-07	0.6746352e-07%	1.200000e-05	1.200000e-05	3.495253e+17	0.536346e-07	0.6746352e-07%
flash	8388608	uint8_t	8388608	1	clock()	5.195000e-03	5.195000e-03	1.614746e+15	0.648690e-07	1.7476346e-07%	1.300000e-05	1.300000e-05	6.452775e+17	0.648690e-07	1.7476346e-07%
flash	12582912	uint8_t	12582912	1	clock()	7.637000e-03	7.637000e-03	1.647625e+15	1.675687e-07	2.8574463e-07%	4.500000e-05	4.500000e-05	2.796203e+17	1.675687e-07	2.8574463e-07%
flash	16777216	uint8_t	16777216	1	clock()	1.028700e-02	1.028700e-02	1.630914e+15	1.768674e-07	2.6536463e-07%	1.400000e-05	1.400000e-05	1.198373e+18	1.768674e-07	2.6536463e-07%
flash	20971520	uint8_t	20971520	1	clock()	1.280500e-02	1.280500e-02	1.637760e+15	1.436346e-07	1.9636346e-07%	1.200000e-05	1.200000e-05	1.747627e+18	1.436346e-07	1.9636346e-07%
flash	25165824	uint8_t	25165824	1	clock()	1.510000e-02	1.510000e-02	1.666611e+15	2.646336e-07	3.8578255e-07%	4.600000e-05	4.600000e-05	5.470831e+17	2.646336e-07	3.8578255e-07%
flash	29360128	uint8_t	29360128	1	clock()	1.747400e-02	1.747400e-02	1.680218e+15	2.565474e-07	4.8968455e-07%	2.800000e-05	2.800000e-05	1.048576e+18	2.565474e-07	4.8968455e-07%
flash	33554432	uint8_t	33554432	1	clock()	1.958300e-02	1.958300e-02	1.713447e+15	2.857855e-07	3.9678588e-07%	3.600000e-05	3.600000e-05	9.320676e+17	2.857855e-07	3.9678588e-07%
flash	37748736	uint8_t	37748736	1	clock()	2.203000e-02	2.203000e-02	1.713515e+15	1.789685e-07	4.6536463e-07%	2.800000e-05	2.800000e-05	1.348169e+18	1.789685e-07	4.6536463e-07%
flash	41943040	uint8_t	41943040	1	clock()	2.508800e-02	2.508800e-02	1.671837e+15	2.657447e-07	3.6536463e-07%	3.100000e-05	3.100000e-05	1.353001e+18	2.657447e-07	3.6536463e-07%
flash	46137344	uint8_t	46137344	1	clock()	2.761100e-02	2.761100e-02	1.670977e+15	2.976857e-07	4.6536463e-07%	3.800000e-05	3.800000e-05	1.214141e+18	2.976857e-07	4.6536463e-07%
flash	50331648	uint8_t	50331648	1	clock()	3.138600e-02	3.138600e-02	1.603634e+15	1.946744e-07	1.3985463e-07%	1.300000e-05	1.300000e-05	3.871665e+18	1.946744e-07	1.3985463e-07%
flash	54525952	uint8_t	54525952	1	clock()	3.173400e-02	3.173400e-02	1.718219e+15	1.433535e-07	1.6536463e-07%	1.300000e-05	1.300000e-05	4.194304e+18	1.433535e-07	1.6536463e-07%
flash	58720256	uint8_t	58720256	1	clock()	3.422600e-02	3.422600e-02	1.715662e+15	1.963636e-07	2.9842984e-07%	1.200000e-05	1.200000e-05	4.893355e+18	1.963636e-07	2.9842984e-07%
flash	62914560	uint8_t	62914560	1	clock()	3.702800e-02	3.702800e-02	1.699108e+15	1.436346e-07	3.5863636e-07%	1.100000e-05	1.100000e-05	5.719505e+18	1.436346e-07	3.5863636e-07%
flash	67108864	uint8_t	67108864	1	clock()	4.117200e-02	4.117200e-02	1.629964e+15	1.465346e-07	1.6536463e-07%	1.400000e-05	1.400000e-05	4.793490e+18	1.465346e-07	1.6536463e-07%
flash	71303168	uint8_t	71303168	1	clock()	4.183900e-02	4.183900e-02	1.704227e+15	1.678646e-07	0.6455636e-07%	1.200000e-05	1.200000e-05	5.941931e+18	1.678646e-07	0.6455636e-07%
flash	75497472	uint8_t	75497472	1	clock()	4.908200e-02	4.908200e-02	1.538191e+15	1.787895e-07	0.4372842e-07%	1.000000e-05	1.000000e-05	7.549747e+18	1.787895e-07	0.4372842e-07%
flash	79691776	uint8_t	79691776	1	clock()	5.763200e-02	5.763200e-02	1.382770e+15	2.657447e-07	0.2423525e-07%	1.500000e-05	1.500000e-05	5.312785e+18	2.657447e-07	0.2423525e-07%
flash	83886080	uint8_t	83886080	1	clock()	5.881600e-02	5.881600e-02	1.426246e+15	2.976857e-07	0.6356463e-07%	1.100000e-05	1.100000e-05	7.626007e+18	2.976857e-07	0.6356463e-07%

Таблица. 2. Оценка SSD и flash

Видим, что:

Пропускная способность SSD при чтении/записи (Read/Write) блока размером 4194304 бит — 2.6 * 10^15 Mb/s и 1.3 * 10^15 Mb/s, а при максимальном размере обрабатываемых данных — скорость чтения гораздо выше скорости записи.

Пропускная способность флеш-накопителя при чтении/записи ниже, чем у SSD, однако растет по мере увеличения объема обрабатываемых данных.

Построены диаграммы.

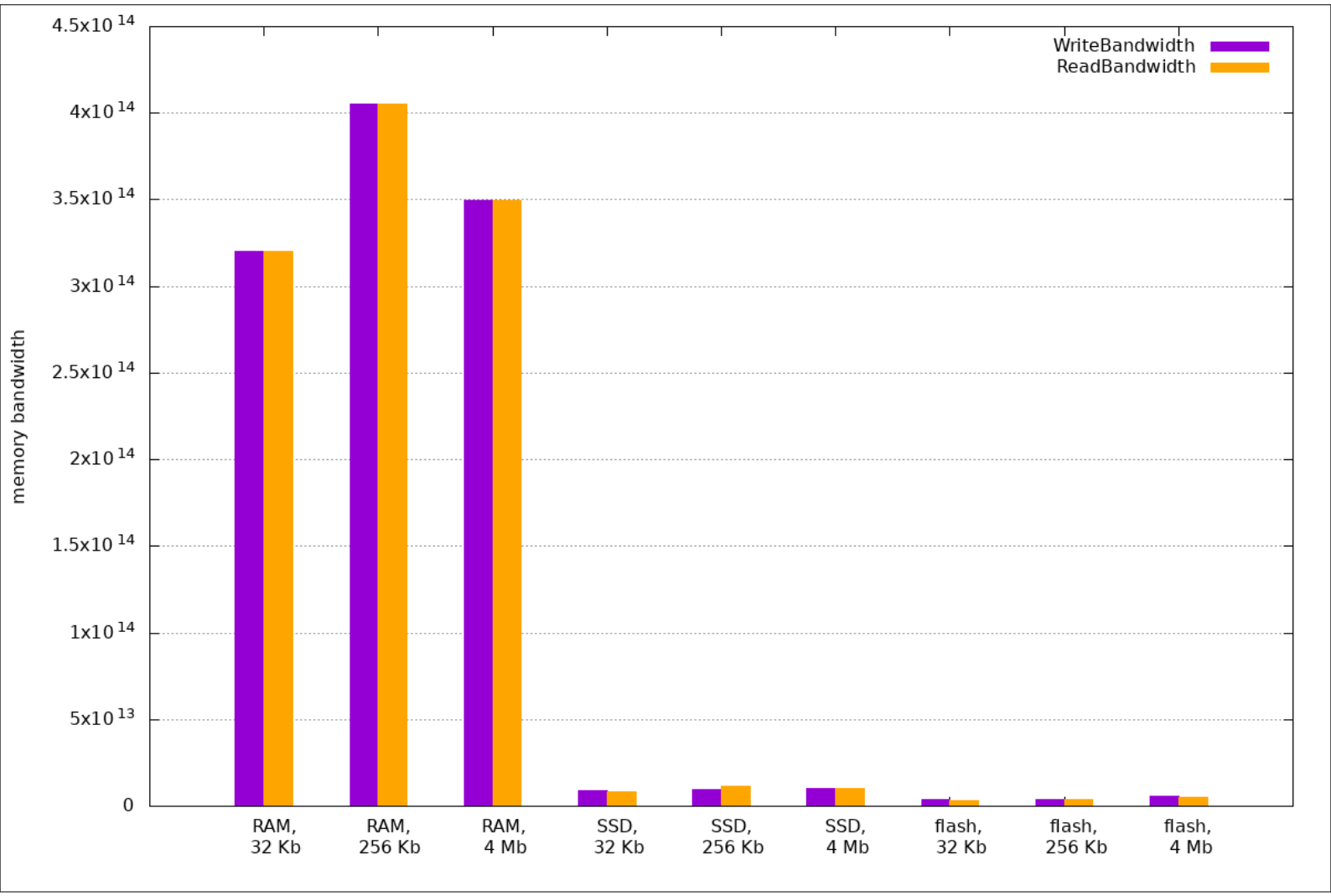

График. 1. Write/ReadBandWidth

На Рис. 3. видим пропускную способность (Мб/с) каждого вида памяти в зависимости от объема данных (32 Кб, 256 Кб и 4 Мб). Данный флеш — накопитель имеет физически низкую скорость чтения/записи — результат ожидаемый.

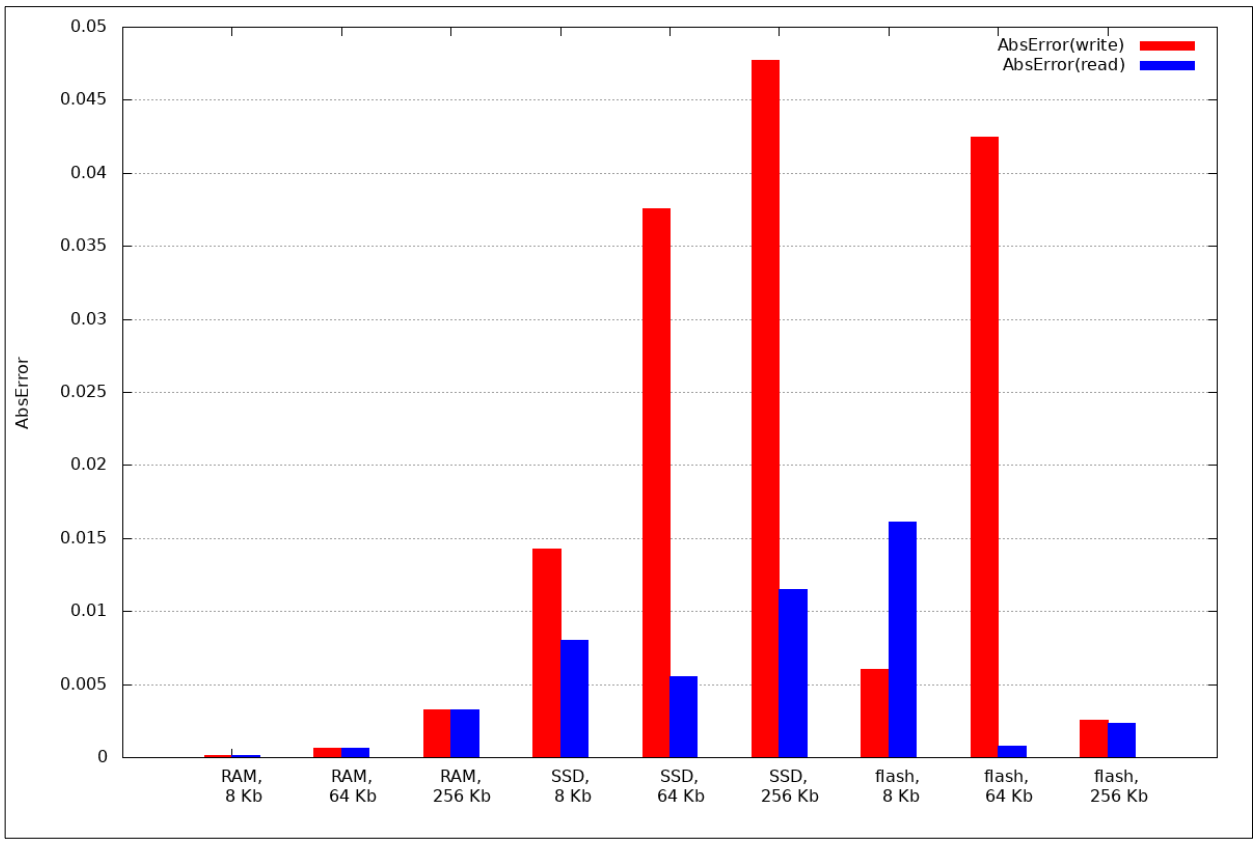

График. 2. AbsError

Видим, что абсолютная погрешность измерения времени максимальна при записи блока данных размером 256 Кб на твердотельном накопителе.

Абсолютная погрешность измерения времени достигает значительно больших показателей в ряде испытаний, как с SSD накопителем, так и с flash.

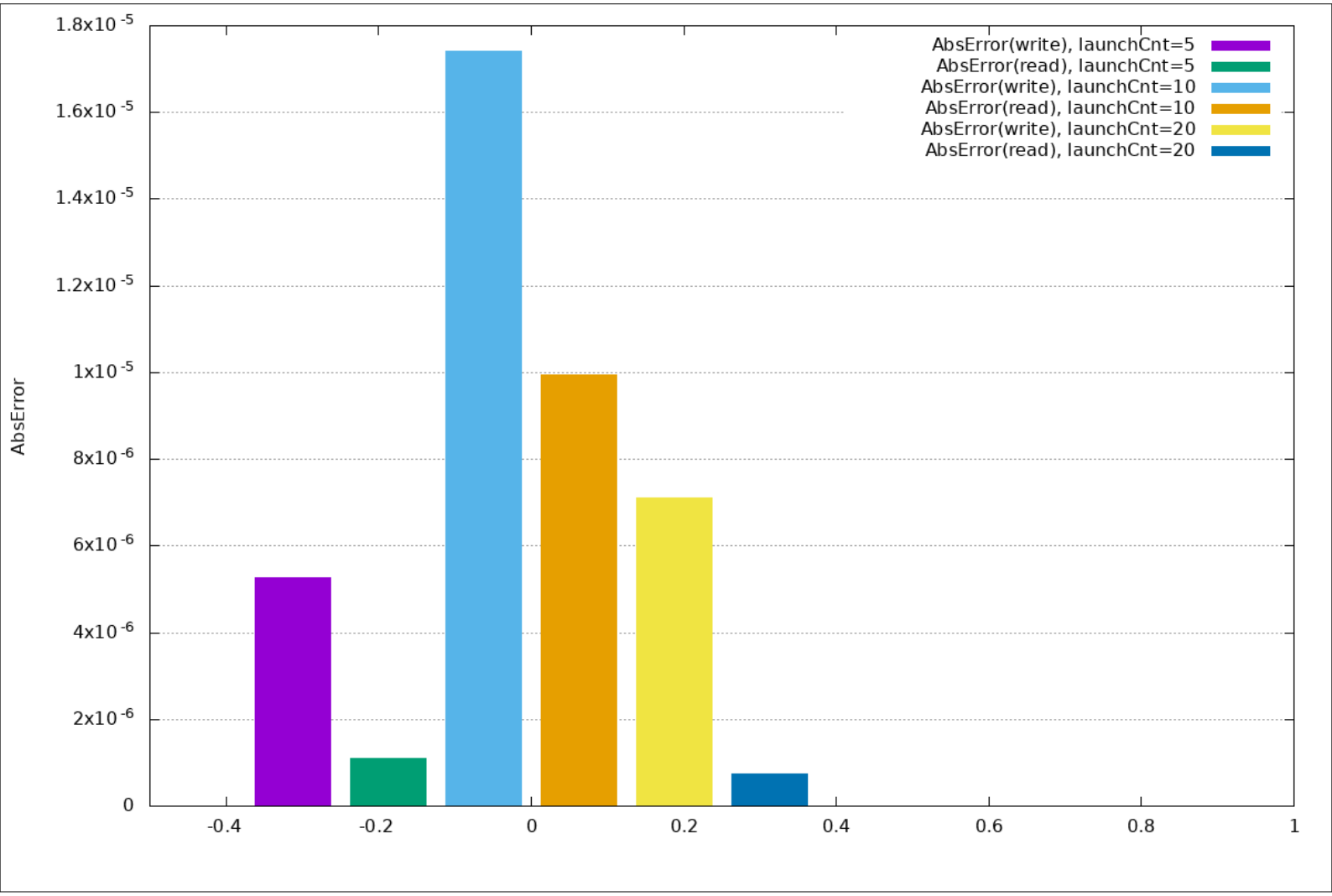

График. 3. AbsError(launchCnt)

На данной диаграмме видим полученное значение абсолютной погрешности (AbsError) чтения и записи для испытаний с 5, 10, 20 итерациями.

Листинг

main.cpp

```
#include <cmath>
#include <cstring>
#include <fstream>
#include <iostream>
#include <string>
#include <sys/time.h>

using namespace std;

double wtime() {
    struct timeval t;
    gettimeofday(&t, NULL);
    return (double) t.tv_sec + (double) t.tv_usec * 1E-6;
}

double getAverage(const double *array, int size) {
    double average = 0;
    for (int i = 0; i < size; i++) {
        average += array[i];
    }
    return average / size;
}

long long getSize(string block) {
    long long size = 1, _size;
    bool isNum = true;
    long long i;
    string s = "";
    for (i = 0; i < block.size(); i++) {
        if (!isdigit(block[i])) {
            isNum = false;
            break;
        }
        s.push_back(block[i]);
    }
    if (block[i] == 'K' && block[i + 1] == 'b') {
        size = 1024;
    } else if (block[i] == 'M' && block[i + 1] == 'b') {
        size = 1048576;
    }

    return static_cast<long long>(stoi(s)) * size;
}

void initArray(char *array, long long size) {
    for (long long i = 0; i < size; i++) {
        array[i] = rand() % 256;
    }
}
```

```
double maxElem(double *timeBuf, int size) {
    double max = timeBuf[0];
    for (int i = 1; i < size; i++) {
        if (max < timeBuf[i]) {
            max = timeBuf[i];
        }
    }
    return max;
}

void writeCSV(int mode, long long blockSize, double *writeTimeBuf, double avWriteTime, double
*readTimeBuf, double avReadTime, int launchCount) {
    string memoryType;
    if (mode == 1)
        memoryType = "RAM";
    else if (mode == 2)
        memoryType = "HDD";
    else if (mode == 3)
        memoryType = "SSD";
    else if (mode == 4)
        memoryType = "flash";
    string elementType = "char";
    long long bufferSize = blockSize;
    int launchNum = launchCount;
    double writeTime = round(writeTimeBuf[launchNum] * 100) / 100;
    string timer = "sys/time.h";
    string writeBandWidth = to_string(round(blockSize / avWriteTime * 100) / (100 * 1024 *
1024));
    double absErrorWrite = round((maxElem(writeTimeBuf, launchCount) - avWriteTime) * 100) / 100;
    double relErrorWrite = absErrorWrite / avWriteTime * 100;
    double readTime = round(readTimeBuf[launchNum] * 100) / 100;
    string readBandWidth = to_string(round(blockSize / avReadTime * 100) / (100 * 1024 * 1024));
    double absErrorRead = round((maxElem(readTimeBuf, launchCount) - avReadTime) * 100) / 100;
    double relErrorRead = absErrorRead / avReadTime * 100;

    ofstream fout("file.csv", ios_base::app);
    if (fout.good()) {
        fout << memoryType << ";" << blockSize << ";" << elementType << ";" << bufferSize << ";"
             << launchNum << ";" << timer << ";" << writeTime << ";" << avWriteTime << ";"
             << writeBandWidth << ";" << absErrorWrite << ";" << relErrorWrite << "%;"
             << readTime << ";" << avReadTime << ";" << readBandWidth << ";"
             << absErrorRead << ";" << relErrorRead << "%\n";
    }
    fout.close();
}

double noEchoRead(char *array, long long size, const string &path = "", int mode = 1) {
    double time = -1;
    if (mode == 1) {
        time = wtime();
        for (long long i = 0; i < size; i++) {
            *(array + i);
        }
        time = wtime() - time;
    } else {
        ifstream fin(path);
```

```
        if (fin.good()) {
            time = wtime();
            while (!fin.eof()) {
                fin.get();
            }
            time = wtime() - time;
        }
    }

    return time;
}

pair<double, double> testMemory(long long size, int mode, const string& path = "") {
    double timeRead = -1;
    double timeWrite = -1;
    char *buffer = new char[size];

    if (mode == 1) {
        timeWrite = wtime();
        for (long long i = 0; i < size; i++) {
            buffer[i] = static_cast<char>(rand() % 26 + 65);
        }
        timeWrite = wtime() - timeWrite;
        timeRead = noEchoRead(buffer, size);
    } else if (mode > 1 && mode < 5) {
        ofstream fout(path);
        if (fout.good()) {
            initArray(buffer, size);
            timeWrite = wtime();
            for (long long i = 0; i < size; i++) {
                fout << buffer[i];
            }
            timeWrite = wtime() - timeWrite;
        }
        fout.close();
        timeRead = noEchoRead(buffer, size, path, mode);
    }
    return make_pair(timeWrite, timeRead);
}

int main(int argc, char **argv) {
    srand(time(NULL));

    int launchCount = 0;
    int mode;
    long long size = 0;
    string path;

    if (argc != 9) {
        cout << "./memory_test –m [RAM/HDD/SSD/flash] –b [size(b/Kb/Mb (b default))] –l [launch
count] -p [path]" << endl;
        return 1;
    } else {
        for (int i = 1; i < 9; i += 2) {
            if (strcmp(argv[i], "-m") == 0) {
                if (strcmp(argv[i + 1], "RAM") == 0)
```

```
                    mode = 1;
                else if (strcmp(argv[i + 1], "HDD") == 0)
                    mode = 2;
                else if (strcmp(argv[i + 1], "SSD") == 0)
                    mode = 3;
                else if (strcmp(argv[i + 1], "flash") == 0)
                    mode = 4;
            } else if (strcmp(argv[i], "-b") == 0) {
                string strSize = argv[i + 1];
                size = getSize(strSize);
            } else if (strcmp(argv[i], "-l") == 0)
                launchCount = stoi(argv[i + 1]);
            else if (strcmp(argv[i],"-p") == 0 && mode > 1) {
            path = argv[i + 1];
            path += "/test.txt";
            }
        }
    }

    double *writeTimeBuffer = new double[launchCount];
    double *readTimeBuffer = new double[launchCount];

    for (int i = 0; i < launchCount; i++) {
        pair<double, double> time;
        if (mode != 1) {
            time = testMemory(size, mode, path);
        }
        else {
            time = testMemory(size, mode);
        }
        writeTimeBuffer[i] = time.first;
        readTimeBuffer[i] = time.second;
    }
    double averageWriteTime = getAverage(writeTimeBuffer, launchCount);
    double averageReadTime = getAverage(readTimeBuffer, launchCount);

    writeCSV(mode, size, writeTimeBuffer, averageWriteTime, readTimeBuffer, averageReadTime,
launchCount);
    delete(writeTimeBuffer);
    delete(readTimeBuffer);
    return 0;
}
```