**BC/NW 2021№ 2 (37):2.1**

**РАЗРАБОТКА АВТОМАТИЗИРОВАННОГО КОМПЛЕКСА ТЕСТИРОВАНИЯ ПРОИЗВОДИТЕЛЬНОСТИ КОМПОНЕНТОВ ОПЕРАЦИОННОЙ СИСТЕМЫ**

Абросимов Л.И., Киселев А.В.

На сегодняшний момент актуален переход стратегических объектов страны на отечественные вычислительные комплексы.

В настоящее время на предприятиях используются вычислительные средства иностранных производителей.

Для повышения безопасности функционирования российских информационных систем (ИС) необходима замена зарубежных вычислительных средств (ВС) ИС предприятий, основанная на использовании отечественных вычислительных комплексах (ВК) линейки «Эльбрус».

В данной работе рассмотрены вычислительный комплекс на микропроцессоре МЦСТ R1000 с архитектурой SPARC v9 и ВК на микропроцессоре Intel Core I7 с архитектурой X86, которые функционируют под управлением операционной системы (ОС) «Эльбрус».

Замена ВС ИС предусматривает перенос программного обеспечения (ПО) функционирующих ИС, которые используют ВК на микропроцессоре Intel Core I7, на линейку ВК «Эльбрус», при этом время выполнения ПО в новой ИС должно быть не меньше чем в функционирующей ИС.

Замена ВС ИС предусматривает перенос программного обеспечения (ПО) функционирующих ИС, которые используют ВК на микропроцессоре Intel Core I7, на линейку ВК «Эльбрус», при этом время выполнения ПО в новой ИС должно быть не меньше чем в функционирующей ИС.

Целью исследования является тестирование (в качестве ПО) компонентов операционной системы на разных микропроцессорах (Intel Core I7 и Эльбрус R1000) для оценки параметров производительности и принятия решения об использования другой соответствующей реализации линейки ВК «Эльбрус».

Для достижения поставленной цели предлагается автоматизировать процесс тестирования ОСЭ, для чего необходимо решить следующие задачи:

1) проанализировать требования к модернизации информационной системы предприятия;

2) разработать алгоритм и программу тестирования характеристик производительности ОС «Эльбрус»;

3) провести тестирование ОС на ВК на микропроцессоре Intel Core I7, а затем на ВК на микропроцессоре МЦСТ R1000;

4) провести сравнительный анализ результатов экспериментальной проверки.

**Анализ требований к модернизации информационной системы предприятия**

Так как ПО иностранных разработчиков превалирует на российском рынке, а Россия находится под серьезным политическим давлением, то частью борьбы на международной арене вполне может стать блокировка программных решений, на которые завязаны ключевые процессы наших госкомпанией. К тому же, если софт разработан в другой стране – компанией, которая подчиняется правительству своего государства – всегда есть шанс, что все проходящие через программу данные попадут к третьим лицам, а для госкомпаний это неприемлемо.

Самая важная часть программного обеспечения любого ВК это ОС - комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами ВК и прикладными программами, а с другой стороны – предназначены для управления устройствами ВК, вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надёжных вычислений.

Так компанией МЦСТ была создана, сопровождается и постоянно развивается ОС «Эльбрус». Она основана на базе ядра Linux 2.6.33.  
 ОС «Эльбрус», обеспечивает многозадачный и многопользовательский режимы работы. Для неё разработаны особые механизмы управления процессами, виртуальной памятью, прерываниями, сигналами, синхронизацией, поддержка тегированными вычислениями.

В состав ОС «Эльбрус» входят базовые средства поддержки интерфейса пользователей:

• *Средства поддержки интерфейса командной строки* («консоль»). Обеспечивают оператору возможность работы с ВК в текстовом режиме с помощью набора команд и получения текстовых сообщений от операционной системы и запускаемых приложений;

• *Средства архивации для объединения ряда файлов в единый архив или серию архивов* (в том числе со сжатием данных), что обеспечивает удобство передачи через каналы связи или хранения;

• *Средства разработки ПО*. Обеспечивают процесс разработки и поддержки ПО. Это – ассемблеры, трансляторы, компиляторы, компоновщики (редакторы связей), сборщики, препроцессоры, отладчики, текстовые редакторы, библиотеки подпрограмм, средства управления версиями, средства документирования;

• *Средства планирования заданий* — позволяют указать ОС, какие действия, в какое время и с какой периодичностью необходимо выполнить.

Помимо базовых в интерфейс пользователя введён ряд средств, поддерживающих создание функционального ПО.

Средства поддержки графического пользовательского интерфейса содержат базовые компоненты графической системы Xorg, а также набор различных вспомогательных библиотек, в том числе GTK+ и Qt.

Основой ОС является библиотека Glibc – (GNU C Library) – свободно распространяемая библиотека С, которая обеспечивает системные вызовы и основные функции, такие как open, malloc, printf и т.д. Библиотека C используется для всех динамически скомпонованных программ. Glibc используется в системах, на которых работает много разных ОС, и на разных архитектурах. Наиболее часто Glibc используется на x86-машинах с ОС Linux. Также официально поддерживаются архитектуры SPARC и «Эльбрус».

Библиотека glibc, поставляемая в составе ОС Эльбрус, сформирована на основе GNU glibc версии 2.7. Она состоит из двух частей:

• заголовочные файлы, которые определяют типы и макрокоманды и объявляют переменные и функции;

• фактическая библиотека или архив, который содержит определения переменных и функций. Состоит из нескольких файлов, функции в которых объединены по определенному признаку (например, libm.a – архив математических функций).

Для поддержки программ, работающих в защищённом режиме, поставляется компактная библиотека libmcst, обеспечивающая функции работы с памятью и поддержку ввода-вывода на уровне базовой библиотеки libc.

В ядро ОС «Эльбрус» встроен комплекс средств защиты информации (КСЗИ) от несанкционированного доступа (НСД). Полное функционирование КСЗИ ОС «Эльбрус» должно обеспечивать требуемый уровень защиты информации от НСД при работе ВК в составе специализированных автоматизированных систем. КСЗИ реализуется использованием системных вызовов, библиотек подпрограмм, конфигурированием системы.

КСЗИ от НСД ОС «Эльбрус» предоставляет возможность применять средства вычислительной техники (СВТ) серии «Эльбрус» в составе ВК для построения автоматизированных систем. В этом случае СВТ:

• отвечают требованиям 2-го класса защищённости от   
НСД РД Гостехкомиссии при президенте РФ;

• позволяют проводить сертификацию ОПО СВТ по 2-му уровню контроля не декларированных возможностей, в соответствии с   
РД Гостехкомиссии при президенте РФ.

На данный момент на предприятии для решения задач используются ВК на процессоре Intel Core I7 – 4770, которые функционируют под управлением ОС «Эльбрус». Характеристики процессора приведены в Таблица 1.

Таблица 1 - характеристики процессора Intel Core I7

|  |  |
| --- | --- |
| **Наименование параметра** | **Значение** |
| Архитектура | X86 |
| Количество ядер | 4 |

Продолжение Таблицы 1 - характеристики процессора Intel Core I7

|  |  |
| --- | --- |
| **Наименование параметра** | **Значение** |
| Количество потоков | 8 |
| Базовая тактовая частота процессора | 3.4 GHz |
| Техпроцесс | 22 нм |
| Кэш L1 (инструкции) | 128 Кб |
| Кэш L1 (данные) | 128 Кб |
| Объем кэша L2 | 1 Мб |
| Объем кэша L3 | 8 Мб |
| Год начала производства | 2013 |

Для повышения безопасности функционирования информационной системы необходима замена вычислительных средств ИС предприятия, основанная на использовании отечественных вычислительных комплексах линейки «Эльбрус».

Для использования ВК серии «Эльбрус» проделана фундаментальная работа по преобразованию ОС Linux в операционную систему, поддерживающую режим работы в реальном времени, для чего были реализованы актуальные оптимизации в ядре, и затем на базе стандартной библиотеки управления потоками вычислений и синхронизацией libpthread была создана собственная оптимизированная библиотека elpthread. В ходе работы в реальном времени можно устанавливать различные режимы обработки внешних прерываний, планирования вычислений, обменов с дисковыми накопителями и некоторые другие.

В настоящей cтатье рассматривается ВК на микропроцессоре МЦСТ R1000. Основные технические характеристики процессора приведены в Таблица 2.

Таблица 2 - характеристики процессора МЦСТ R1000

|  |  |
| --- | --- |
| **Наименование параметра** | **Значение** |
| Архитектура | SPARC V9 |
| Количество ядер | 4 |
| Количество потоков | 4 |
| Базовая тактовая частота процессора | 1 GHz |
| Техпроцесс | 90 нм |
| Кэш L1 (команды) | 16 Кб |
| Кэш L1 (данные) | 32 Кб |
| Объем кэша L2 | 2 Мб |
| Пропускная способность канала межпроцессорного обмена (дуплекс) | 3х4 Гбайт/c |
| Год начала производства | 2011 |

Для принятия решения об использовании другой линейки ВК «Эльбрус» необходимо провести тестирование компонентов ОС на разных микропроцессорах для оценки параметров производительности.

Тестирование характеристик производительности ОС «Эльбрус» будет проводиться с помощью готовых пакетов тестирований: [X]

* Lmbench;
* Rt-tests;
* Iozone;
* Unixbench.

**Постановка задачи разработки автоматизированного комплекса тестирования**

1. Постановка задачи разработки автоматизированного комплекса тестирования

В отрасли, которая зависит от производительности, легко сосредоточиться на скорости процессора или количестве ядер на сокет. Однако производительность памяти в равной степени важна для общей производительности системы. Если процессор не соответствует правильной памяти, это может привести к недостаточному использованию процессора, что приведет к снижению эффективности и снижению общей производительности. Существует два ключевых фактора производительности, с помощью которых компьютерная система извлекает данные. Задержка памяти и пропускная способность играют огромную роль в производительности и обработке данных.

Задержка и пропускная способность являются важными измеряемыми параметрами. Эти параметры являются хорошими индикаторами, которые описывают влияние во время процессов настройки, сравнивают данные одной платформы с другой и, возможно, определяют другие потенциальные области повышения производительности.

Для измерения пропускной способности используется программа bw\_mem, а для измерения латентности памяти используется программа lat\_mem\_rd, которые являются частью пакета тестирования lmbench.

Lmbench — это набор программ, написанных Ларри МакВоем (Larry McVoy. Пакет lmbench универсален и может работать на множестве операционных систем семейства Unix. Это объясняется тем, что программы-тесты написаны в соответствии со стандартом ANSI C и используют системные вызовы. Первое означает, что они могут быть скомпилированы любым из множества компиляторов, поддерживающих упомянутый стандарт. Второе говорит о том, что коль скоро программный интерфейс системных вызовов в различных операционных системах, отвечающих стандарту POSIX, является одинаковым, lmbench позволяет сравнить производительность различных операционных систем данного семейства.

Lmbench позволяет протестировать все основные элементы вычислительной системы: процессор, аппаратный кэш, оперативную память, дисковый ввод-вывод, сетевую подсистему, а также механизмы операционной системы: время создания и переключения процессов, передачу данных через каналы, обработку сигналов, выполнение системных вызовов и другое.

Основные особенности LMbench:

• Портативность тестирования для операционных систем

Инструменты оценки написаны на C и обладают хорошей переносимостью. Это полезно для получения подробных результатов сравнения между системами.

• Результаты расчета базы данных

Результаты расчета базы данных включают результаты работы большинства основных производителей компьютерных рабочих станций.

• Результат расчета задержки памяти

Тест задержки памяти показывает задержку кэша всех систем (данных), таких как кэш уровня 1, уровня 2 и уровня 3, а также пропуск памяти и таблицы TLB. Кроме того, размер кэша может быть правильно разделен на несколько наборов результатов и считан.

• Результат расчета конверсии контекста

Этот инструмент оценки не обращает особого внимания на цитирование только количества «в кеше». Он часто изменяется между количеством и размером процессов, и когда текущий контент отсутствует в кэше, результаты делятся таким образом, чтобы это было видно пользователю.

• Регрессионное тестирование

Sun и SGI использовали этот инструмент оценки для поиска и устранения проблем с производительностью.

Intel использовала их во время разработки P6.

Linux использует их для настройки производительности Linux.

• Новые инструменты оценки

Исходный код является относительно небольшим, читаемым и легко расширяемым. Как обычно, его можно объединить в разные формы для проверки другого контента.

Bw\_mem выделяет вдвое больший объем памяти, обнуляет его, а затем умножает копирование первой половины на вторую половину. Результаты отображаются в мегабайтах, перемещаемых в секунду.

Бенчмарк работает как два вложенных цикла. Внешний – по размерам шагов, внутренний – по размерам массива. При этом для каждого размера массива тест создает кольцо пойнтеров, указывающих на один шаг назад.

Еще одним параметром, который влияет на производительность системы, является время переключения контекста. В пакете Lmbench за это отвечает программа lat\_ctx. Она измеряет время переключения контекста для любого разумного количества процессов любого разумного размера. Процессы связаны в кольцо каналов Unix. Каждый процесс читает токен из своего канала, возможно, выполняет некоторую работу, а затем записывает токен в следующий процесс (Рисунок 1).

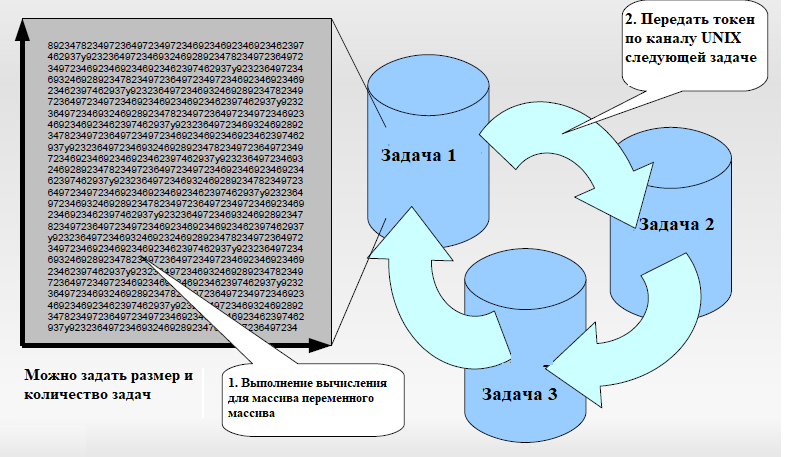


Рисунок 1 - принцип работы теста lat\_ctx

Процессы могут различаться по размеру. Нулевой размер — это базовый процесс, который ничего не делает за исключением передачи токена следующему процессу. Размер процесса больше нуля означает, что процесс выполняет некоторую работу перед передачей токена. Работа моделируется как суммирование массива указанного размера. Суммирование представляет собой развернутый цикл примерно 2,7 тысячи инструкций.

Измерительные потоки периодически пробуждаются с заданным интервалом по истечению таймера (циклически аварийный сигнал). Затем вычисляется разница между запрограммированным и эффективным временем пробуждения и передается главному потоку через общую память. Главный поток отслеживает значения задержки и печатает минимальную, максимальную и среднюю задержки.

Наиболее важным значением в результатах является максимальная обнаруженная задержка, поскольку это значение может дать представление о длине задержки наихудшего случая в оцениваемой ситуации.

Для тестирования файловой системы можно воспользоваться бенчмарком Iozone. Он генерирует и измеряет различные файловые операции. Iozone был портирован на многие машины и работает под многими операционными системами.

Iozone полезен для определения широкого анализа файловой системы компьютерной платформы. Тест тестирует производительность ввода-вывода файлов для следующих операций: чтение, запись, повторное чтение, повторная запись, чтение в обратном порядке и др.

Еще одним полезным бенчмарком, который дает оценку производительности системы является Unixbench. Его представили инженеры из университета Монаша в 1983 году.

Unix / GNU Linux система по результатам тестов получает набор баллов, сравниваемых с баллами тестов эталонной системы (имеющей рейтинг 10.0 баллов), сейчас как эталон используется рабочая станция SPARCstation 20-61 (основанная на микропроцессорах SuperSPARC и hyperSPARC).

Сравнение результатов тестов unixbench с оценками эталонной системы позволяют получить "индексы", которые намного легче обрабатывать чем "сырые" оценки производительности. По окончании тестирования весь набор "индексов" объединяется, для создания общего "индекса" для системы.

Для автоматизации процесса тестирования разработна программа, которая будет включать данные бенчмарки и позволит протестировать систему сначала на ВК с микропроцессором Intel, а затем на ВК с микропроцессором МЦСТ.

**2. Разработка алгоритма и программы для тестирования характеристик производительности ОС «Эльбрус»**

Алгоритм автоматизированного комплекса тестирования представлен на Рисунке 2.

1. Запуск программы; исходные данные для работы алгоритма:

* для бенчмарка bw\_mem — это количество повторений (по умолчанию 1), параллелизм (по умолчанию 1), размер и вариант тестирования;
* для бенчмарка lat\_mem\_rd — это количество повторений (по умолчанию 1), параллелизм (по умолчанию 1), размер памяти и размер шага (по умолчанию 128);
* для бенчмарка lat\_ctx — это количество повторений (по умолчанию 1), параллелизм (по умолчанию 1), размер (по умолчанию 0 кб) и количество процессов;
* для бенчмарка cyclictest — это количество циклов (по умолчанию 0 (бесконечно)), количество потоков (по умолчанию 1), период пробуждения потока (по умолчанию 1 мс), расстояние между потоками (по умолчанию 500);
* для бенчмарка hackbench — это количество групп (по умолчанию 10), количество файловых дескрипторов в каждой группе (по умолчанию 40), количество сообщений (по умолчанию 100) и размер сообщения (по умолчанию 100 байт);
* для бенчмарка Iozone — по умолчанию бенчмарк запускается в автоматическом режиме, т.е. выполняет все возможные тесты;
* для бенчмарка Unixbench — по умолчанию бенчмарк выполняет все тесты с одной копией, а затем все тесты с N копиями, где N – это количество процессоров;

1. выбор функционала программы;
2. просмотреть результаты тестирований? Если да, то п. 12),  
   иначе п. 4);
3. провести тестирование характеристик производительности  
   ОС «Эльбрус»? Если да, то п. 7), иначе п. 5);
4. выйти из программы? Если да, то п. 6), иначе п. 2);
5. завершение программы;
6. выбор бенчмарка для тестирования характеристик производительности ОС;
7. провести тестирование с помощью пакета тестов Lmbench? Если да, то п. 14), иначе п. 9);
8. провести тестирование с помощью пакета тестов Rt-tests? Если да, то п. 15), иначе п. 10);
9. провести тестирование с помощью бенчмарка Iozone? Если да,  
   то п. 16), иначе п. 11);
10. провести тестирование с помощью бенчмарка Unixbench? Если да, то п. 17), иначе п. 23);
11. поиск результата предыдущего тестирования в зависимости от выбранного теста;
12. вывод результата тестирования на экран. Далее п. 18);
13. выбор программы тестирования из пакета Lmbench;
14. выбор программы тестирования из пакета Rt-tests;
15. запуск бенчмарка Iozone. Используется для тестирования файловой системы. В качестве дополнительных параметров задаются:

* включение Excel режима;
* сохранение вывода в электронную таблицу;
* тестирование всех возможных размеров записей;
* запуск определенного теста;
* создание файлов смещения/задержек;
* установки минимального/максимального размера файла;
* включение использования ЦП;

1. запуск бенчмарка Unixbench. Используется для предоставления информации о базовых параметрах производительности Unix-подобных систем при выполнении одной или нескольких задач. В качестве дополнительных параметров задаются:

* запуск в тихом режиме;
* запуск в подробном режиме;
* количество итераций для каждого теста;
* запуск n копий каждого теста параллельно;

1. выйти из поиска результатов предыдущего тестирования, если да, то п.2), иначе п.12);
2. провести тестирование с помощью бенчмарка cyclictest   
   (без нагрузки)? Если да, то п. 24), иначе п. 20);
3. провести тестирование с помощью бенчмарка cyclictest   
   (с нагрузкой (hackbench))? Если да, то п. 25), иначе п. 7);
4. сбор полученных данных;
5. сохранение данных на диск;
6. завершение процесса тестирования? Если да, то п. 2), иначе п. 7);
7. запуск cyclictest (без нагрузки). Это один из наиболее часто используемых инструментов для оценки относительной производительности систем реального времени. В качестве дополнительных параметров задаются:

* приоритет;
* блокирование текущего и будущего выделения памяти;
* использование clock\_nanosleep вместо таймеров интервала posix;
* печать отчета на выходе;
* параметр для тестирования в системах SMP;

1. запуск cyclictest (с нагрузкой (hackbench)). Для создания нагрузки на систему при измерении задержек с помощью cyclictest используется программа hackbench.
2. провести тестирование с помощью бенчмарка bw\_mem? Если да, то п. 29), иначе п. 27);
3. провести тестирование с помощью бенчмарка lat\_mem\_rd? Если да, то п. 30), иначе п. 28);
4. провести тестирование с помощью бенчмарка lat\_ctx? Если да, то п. 30), иначе п. 7);
5. запуск bw\_mem. Используется для тестирования пропускной способности памяти. Входные данные – количество повторений, параллелизм, размер и вариант тестирования;
6. запуск lat\_mem\_rd. Используется для измерения латентности памяти при чтении. Входные данные - количество повторений, параллелизм, размер памяти и размер шага;
7. запуск lat\_ctx. Используется для измерения времени переключения контекста для любого разумного количества процессов любого разумного размера. Входные данные - количество повторений, параллелизм, размер и количество процессов.

Блоки 16, 17, 24, 25, 29, 30, 31 являются уже готовыми скриптами-тестами, поэтому происходит вызовы процессов, которые выполняют эти скрипты

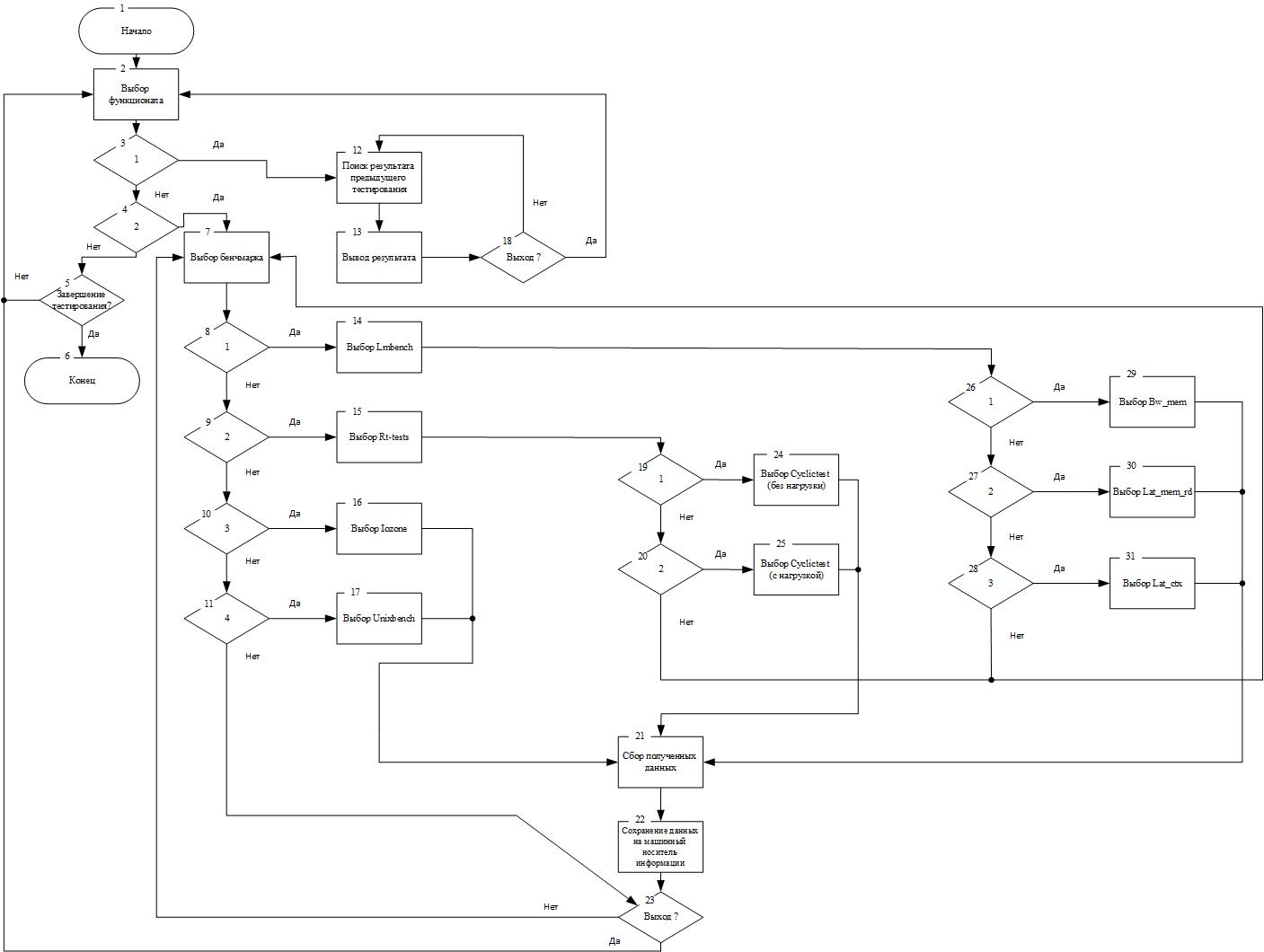


Рисунок 2 Алгоритм автоматизированного комплекса тестирования

Для разработки программного обеспечения, позволяющего автоматизировать процесс тестирования операционной системы, были выбраны язык программирования C++ и интегрированная среда разработки Qt.

Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой системы без изменения исходного кода.

Проведение экспериментальной проверки функционирования разработанного алгоритма

Вначале тестирование проводится на вычислительном комплексе с процессором intel core i7-4770 c базовой тактовой частотой 3.4 ГГц, а затем на вычислительном комплексе с процессором МЦСТ R1000 с базовой тактовой частотой 1 ГГц.

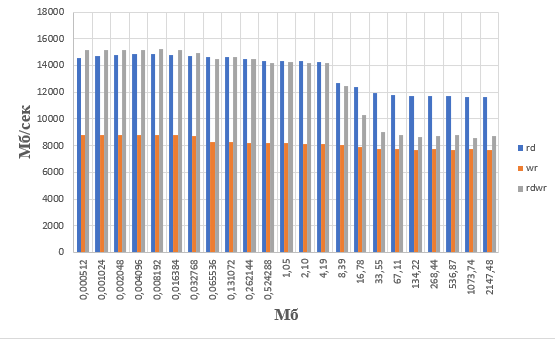


Рисунок 3 - графическое отображение результата бенчмаркаbw\_mem  
 (1 копия) на ВК с процессором Intel

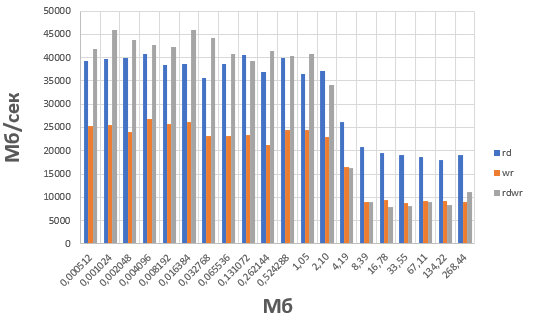


Рисунок 4 - графическое отображение результата бенчмаркаbw\_mem   
(4 копии) на ВК с процессором Intel

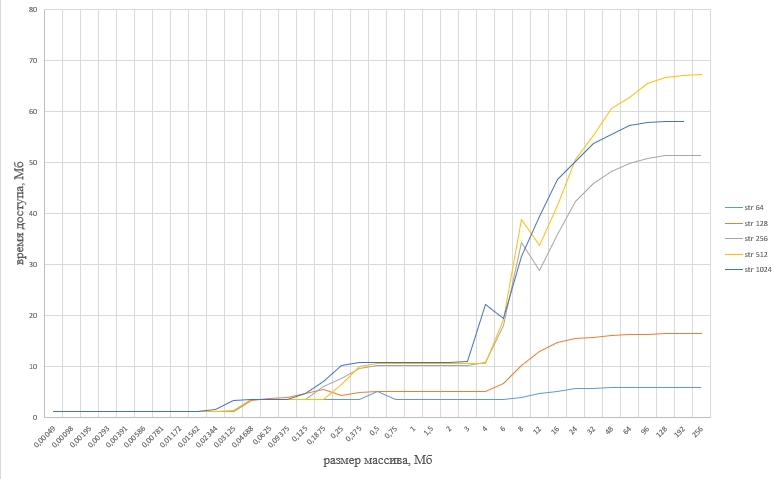


Рисунок 5 - график отображения результата измерения латентности   
(1 копия) на ВК с процессором Intel

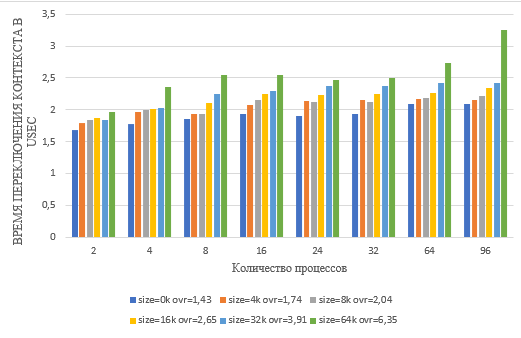


Рисунок 6 - гистограмма отображения результата измерения времени переключения контекста (1 копия) на ВК с процессором Intel

**Проведение тестирования на ВК с микропроцессором МЦСТ**

Сперва cyclictest был запущен без нагрузки и без приоритета на ВК с микропроцессором Intel со следующими параметрами: -c 1, -t 1, -m, -n, -q,  
 -l 1000000, -h 10000, -i 100. Затем был задан наивысший приоритет -p 99.

Результаты работы бенчмарка без нагрузки и без приоритета:

* количество итераций: 1000000;
* минимальная задержка: 30;
* среднее значение: 63;
* максимальная задержка: 2727.

Результаты работы бенчмарка без нагрузки и с приоритетом:

* количество итераций: 1000000;
* минимальная задержка: 13;
* среднее значение: 13;
* максимальная задержка: 487.

Графическое отображение результата представлено на.рис. 7

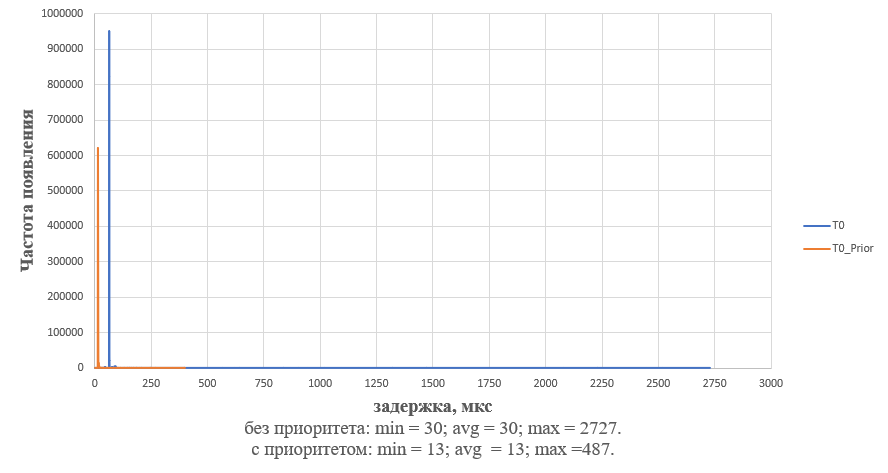


Рисунок 7 -работа бенчмарка cyclictest без нагрузки на 1 потоке на ВК с микропроцессором МЦСТ R1000

Затем cyclictest был запущен с нагрузкой на систему с теми же параметрами. Нагрузка была оказана с помощью бенчмарка hackbench с параметрами: -P, -g 30, -f 100, -s 100, -l 100.

Результаты работы бенчмарка с нагрузкой и без приоритета:

* количество итераций: 1000000;
* минимальная задержка: 40;
* среднее значение: 1781664;
* максимальная задержка: 7279857;

Графическое отображение данного результата невозможно ввиду того, что максимальное значение параметра -h = 1000000, а значения задержек выходят за его границу.

Результаты работы бенчмарка с нагрузкой и с приоритетом:

* количество итераций: 1000000;
* минимальная задержка: 11;
* среднее значение: 18;
* максимальная задержка: 60;

Графическое отображение результата представлено на рис. 8.

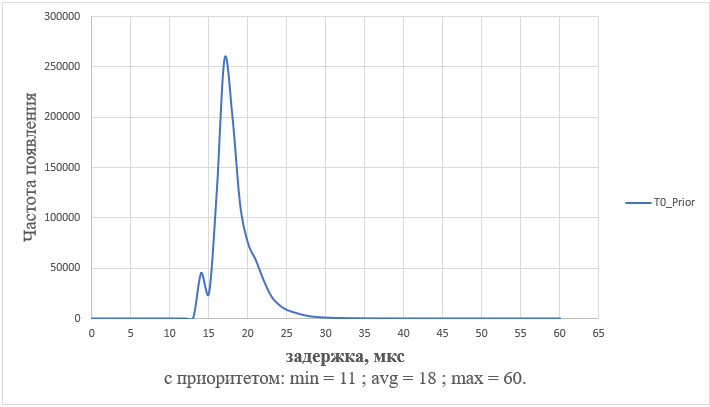


Рисунок 8 -работа бенчмарка cyclictest с нагрузкой и с приоритетом на 1 потоке на ВК с микропроцессором МЦСТ R1000

Далее cyclictest был запущен без нагрузки и без приоритета на ВК с микропроцессором МЦСТ со следующими параметрами: -c 1, -S, -n, -q,  
 -l 1000000, -h 40000, -i 100, -m. Затем был задан наивысший приоритет -p 99.

Результаты работы бенчмарка без нагрузки и без приоритета:

* количество итераций: 1000000, 999924, 999663, 999408;
* минимальная задержка: 31, 63, 63, 37;
* среднее значение: 63, 63, 63, 63;
* максимальная задержка: 2785, 327, 3021, 1996.

Графическое отображение результата представлено на Рисунок 33.

Результаты работы бенчмарка без нагрузки и с приоритетом:

* количество итераций: 1000000, 999831, 999570, 999316;
* минимальная задержка: 12, 13, 13, 13;
* среднее значение: 13, 13, 13, 13;
* максимальная задержка: 31, 35, 28, 21.

Графическое отображение результата представлено на рисунке 9.

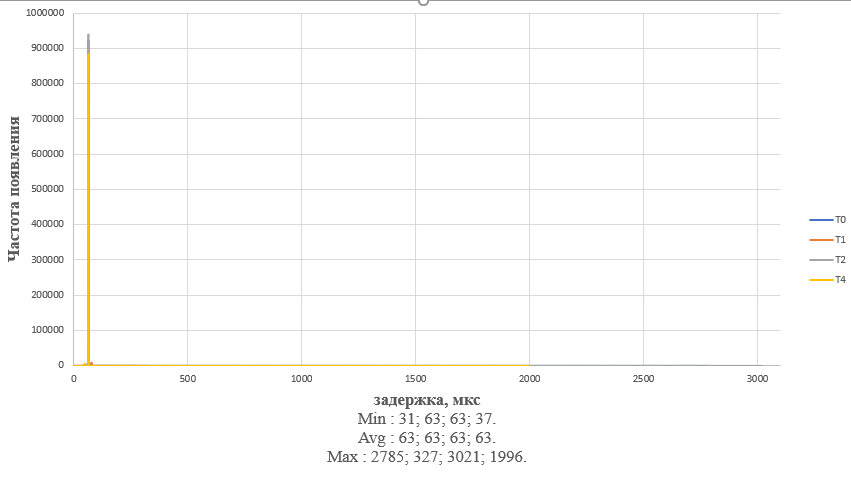


Рисунок 9 -работа бенчмарка cyclictest без нагрузки на 4 потоках без приоритета на ВК с микропроцессором МЦСТ R1000

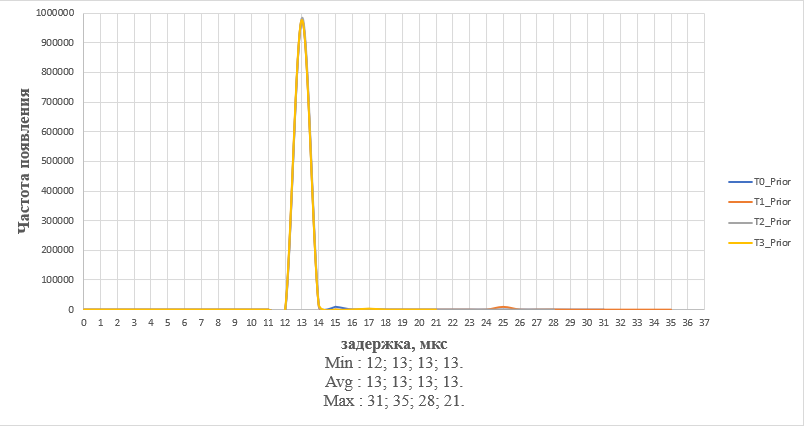


Рисунок 10 -работа бенчмарка cyclictest без нагрузки на 4 потоках с приоритетом на ВК с микропроцессором МЦСТ R1000

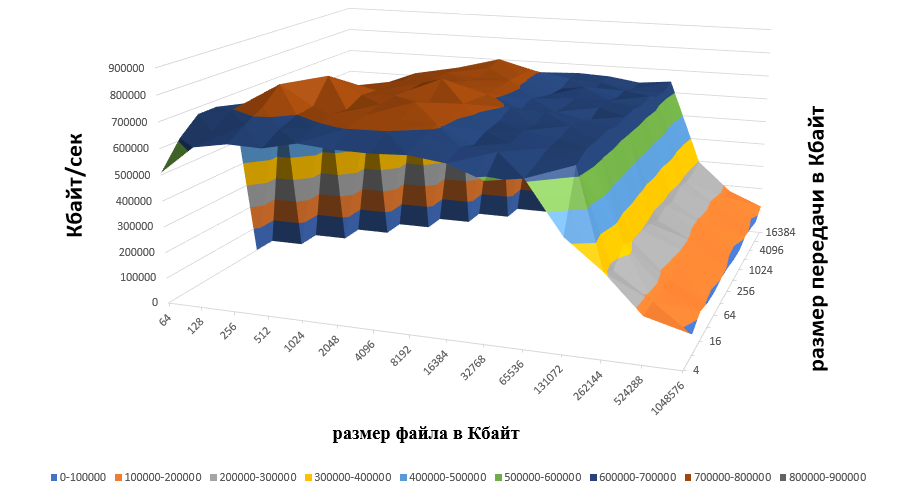


Рисунок 10 - графическое отображение результата Iozone (запись) на ВК с процессором Intel

Итак, для процессора Intel получаем:

* L1 размер: 32 Кб, задержка (средняя): 1.547 нс, циклов = частота \* задержку = 5.26;
* L2 размер 256 Кб, задержка (средняя): 6.046 нс, циклов= частота \* задержку = 20.56;
* L3 размер 5 Мб, задержка (средняя): 14.158 нс, циклов= частота \* задержку = 48.14.

Для процессора МЦСТ R1000:

* L1 размер: 32 Кб, задержка (средняя): 2.999 нс, циклов = частота \* задержку = 2.99;
* L2 размер 1.5 Мб, задержка (средняя): 25.389 нс, циклов= частота \* задержку = 25.39;

Из этих данных можно заметить, что на процессоре МЦСТ задержка в наносекундах больше почти в 2 раза, чем на процессоре Intel, но при этом число прошедших тактов меньше.

**Список используемой литературы**

1. Система тестирования операционной системы. Руководство оператора. ТВГИ.00474-03 34 01.

2. ГОСТ 19.701-90. Единая система программной документации (ЕСПД). Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.

3. OC «Эльбрус». ТВГИ.00311-05 32 01.

4. The Linux Foundation. – URL: https://wiki.linuxfoundation.org

(дата обращения: 10.05.2021). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

5. OSTechNix - Open Source | Technology | Linux And Unix.

– URL: https://ostechnix.com/ (дата обращения: 10.05.2021). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

6. Ruggiero J. Measuring Cache and Memory Latency and CPU to Memory Bandwidth. Intel Corporation, 2008. – 1-14

7. Iozone Filesystem Benchmark. – URL: https://mirror.unpad.ac.id

(дата обращения 01.06.2021). - Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

8. McVoy L., Staelin C. Lmbench: Portable tools for performance

analysis // Proceedings of the USENIX 1996 Annual Technical Conference. – San Diego, The USENIX Association, 1996. – 279-294.

9. Вычислительные сети, теория и практика. – URL: http://network-journal.mpei.ac.ru/ (дата обращения 22.06.2021). - Режим доступа: для зарегистрир. пользователей. – Текст: электронный.