УДК 004.032.24

О.Ю. Чередникова, канд. техн. наук, доц.,
А.Ю. Иванов, ст. преп.,
Е.В. Коцогуб, студент
ГВУЗ «Донецкий национальный технический университет», Украина
jeketos@mail.ru

Исследование многопоточности с использованием волокон на одно- и многоядерном процессорах

В данной работе рассмотрены принципы организации многопоточности с использованием волокон на одно- и многоядерном процессоре. Сравнивается организация многопоточности на основе потоков и волокон. Исследуется обращение из различных потоков и волокон к общей памяти. Предпринята попытка определения процессорного времени, выделяемого каждому потоку операционной системой. Аргументируются преимущества использования волокон для ускорения работы приложения на многоядерном процессоре.

Ключевые слова: поток, волокно, многоядерный процессор, фибер, процесс, многопоточность, общая память, процессорное время.

Введение

Несмотря на возрастающую с каждым годом производительность процессоров, задача увеличения быстродействия работы приложений не потеряла своей актуальности. Кроме того, пользователи уже давно привыкли запускать параллельно несколько приложений для того чтобы делать несколько дел сразу. При этом сами по себе запускаемые приложения могут работать в рамках одного потока - операционная система сама заботится о распределении времени между всеми запущенными приложениями. Однако, все современные операционные системы способны работать в многопоточном режиме, повышая общую производительность системы за счет эффективного распараллеливания выполняемых потоков. Пока один поток находится в состоянии ожидания, например, завершения операции обмена данными с медленным периферийным устройством, другой может продолжать выполнять свою работу. Если механизм организации многопоточности на основе потоков хорошо известен и используется в практике создания пользовательских приложений, то доступные в Win9x/WinNT фиберы (fiber - волокна) [1], используются существенно реже. Под фиберами понимаются приложения – задачи, управление которыми осуществляется разработчиком, а не системой. При этом создается набор стеков, сохраняются регистры процессора и переключаются контексты при обслуживании запросов. Майкрософт включила в Windows поддержку волокон для облегчения портирования существующих приложений из UNIX в Windows [1].

Описание

Волокно (fiber) — своеобразный контекст выполнения, логически подобный потоку, но это не поток [2]. Потоки используются для распараллеливания задач, в то время как волокна больше подойдут для асинхронных операций ввода-вывода. Волокна реально выполняются в одном физическом потоке. Если в настоящем потоке выполнение прерывается само и в произвольном месте кода по воле операционной системы, то в случае волокон разработчик сам решает, когда и где передать управление в другой участок кода программы.

Существенные для исследования особенности фиберов и потоков заключаются в следующем [3]:

- в многопроцессорной системе потоки могут одновременно выполняться на разных процессорах, многофиберный поток будет выполняться только на одном из них. На остальных процессорах в это время могут выполняться, например, фиберы других потоков того же самого процесса;
- фиберы разных потоков одного процесса никак друг с другом не связаны;
- фиберы занимают меньше места в памяти, чем потоки и имеют меньшие затраты времени на переключение контекстов так как выполняются не по инициативе системы, а по запросу программиста;
- при переключении к фиберу он начинает выполняться не с точки входа, а с того места, где выполнение прервалось.
- при использовании потоков существенный момент – это синхронизация между потоками, при использовании волокон нет нужды заботиться о синхронизации т.к. все волокна выполняются по очереди в одном потоке и разработчик сам решает, когда можно передать управление.
- использование волокон потенциально повышает производительность на одном ядре, т.к. переключения между контекстами выполнения можно расставить в наиболее удобных местах.

Однако стоит помнить и об ограничениях:

1) Волокна не помогут задействовать несколько ядер процессора.

2) Если в коде одного волокна произойдет зависание или вечный цикл – все волокна в этом потоке будут заблокированы.

Исследование поведения волокон в одном потоке

В потоке может быть одно или несколько волокон. Единовременно поток будет выполнять код лишь одного волокна — какого, решает сам программист. Для исследования волокон в одном потоке была создана программа на платформе Microsoft Visual Studio 2010, осуществляющая создание трех волокон в потоке и их переключение между собой.

Приступая к работе с волокнами, прежде всего необходимо преобразовать существующий поток в волокно. Это делает функция *ConvertThreadToFiber*:

PV0ID ConvertThreadToFiber(PVOID pvParam);

Она создает в памяти контекст волокна (размером около 200 байтов), в который входят заголовок цепочки структурной обработки исключения, начальный и конечный адреса стека волокна, регистры процессора, включая указатели стека и команд. Если задан параметр NULL, то преобразовывается текущий поток. Именно так задан этот параметр в разработанной программе.

Создав и инициализировав контекст волокна, программист сопоставляет его адрес с потоком, преобразованным в волокно, и теперь оно выполняется в этом потоке. *ConvertThreadToFiber* возвращает адрес, по которому расположен контекст волокна.

Нет смысла преобразовывать поток в волокно, если не будут созданы дополнительные волокна в том же потоке. Чтобы создать другое волокно, вызывается функция *CreateFiber*:

void* fiber = CreateFiber(0,fiberFunc,Null).

Первый параметр определяет, что максимальный размер стека ограничивается 1 Мб. Если указать ненулевое значение, то для стека будет зарезервирован и передан именно такой объем памяти. Функция *CreateFiber* создает и инициализирует новую структуру, представляющую контекст исполнения волокна. При этом сохраняются начальный и конечный адреса памяти нового стека, а также адрес функции волокна (fiberFunc). Как только функция волокна завершится, поток и все созданные в нем волокна тут же будут уничтожены.

Подобно *ConvertThreadToFiber*, функция *CreateFiber* возвращает адрес контекста исполнения волокна. Но, в отличие от *ConvertThreadToFiber*, исполнение созданного функцией *CreateFiber* волокна не начинается, пока исполняется текущее волокно. Дело в том, что исполняться может только одно волокно потока одновременно. Чтобы запустить новое волокно, вызывается функция *SwitchToFiber* [2].

*SwitchToFiber(*fiber*);*

Применение *SwitchToFiber* — единственный способ выделить волокну процессорное время. Поскольку наш код должен явно вызывать эту функцию в нужные моменты, мы полностью управляем распределением процессорного времени для волокон. Такой вид планирования не имеет ничего общего с планированием потоков. Поток, в рамках которого выполняются волокна, всегда может быть вытеснен операционной системой. Когда поток получает процессорное время, выполняется только выбранное волокно, и никакое другое не получит управление, нока мы сами не вызовем SwitchToFiber.

После вызова данной функции управление переходит к волокну *fiber*, а точнее выполняется поточная функция *FiberFunc*

Задача FiberFunc в предлагаемой работе - наращивание счетчика (Cnt) и переключение на следующее волокно. Когда значение счетчика становится равное 10, программа прекращает свою работу и завершается. Результаты работы программы приведены на рис.1.

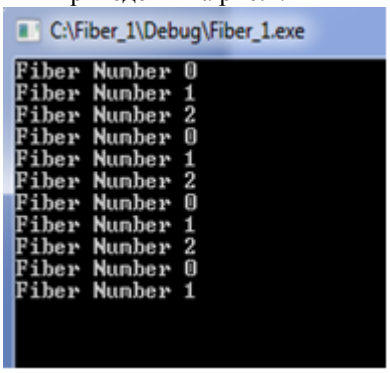

Рисунок 1 - Результаты работы программы переключения волокон в одном потоке

Результаты работы с волокнами показывают возможность программного управления волокнами, переключения их в нужный программисту момент.

Исследование многопоточности волокон на одном ядре

Для исследования многопоточности с использованием волокон на одном ядре процессора и одновременном доступе к разделяемой памяти выбрана следующая последовательность действий:

- создание двух потоков на одном ядре;
- преобразование потока в волокно;
- обращение к разделяемой ячейке памяти (переменной) из разных волокон;

- вывод результатов в массив, а затем в текстовый файл.

В главной процедуре программы создание двух потоков выполняется последовательным вызовом следующих функций:

```
hThread1=CreateThread(NULL,0, Thread1, NULL,
        CREATE_SUSPENDED,&dwIdThread1);
hThread2=CreateThread(NULL,0, Thread2, NULL,
        CREATE_SUSPENDED, &dwIdThread1);
SetThreadPriority(hThread1,
        THREAD_PRIORITY_ABOVE_NORMAL);
SetThreadPriority(hThread2,
        THREAD_PRIORITY_ABOVE_NORMAL);
SetThreadAffinityMask(hTread1, 0x00000001);
SetThreadAffinityMask(hTread2, 0x00000001);
ResumeThread(hThread2);
ResumeThread(hThread1);
```

Функция CreateThread() с [4] параметром CREATE_SUSPEND создает поток в состоянии ожидания и не запускает его до тех пор, пока не будет вызвана функция ResumeThread. Параметр Thread1 задает поточную функцию, которая вызывается сразу после запуска потока. Функция SetThreadPriority устанавливает значение приоритета для заданного потока, а функция SetThreadAffinityMask устанавливает маску, по которой определяется ядро, на котором поток будет выполняться (в данном случае первое). Убедиться в том, что работает первое ядро можно в «Диспетчере задач».

В поточной функции происходит преобразование потока в волокно, создание нового волокна на этом же потоке и переключение на него, как и в случае одного потока.

Для тестирования порядка обращения к волокнам внутри функции каждого волокна организован цикл на 50000 повторений, который инкрементирует глобальную переменную i и записывает ее значение каждое волокно в свой массив.

Результаты тестирования программы показали, что на одноядерном процессоре потоку выделяется определенное процессорное время (квант), по истечении которого управление получает второй поток и, соответственно фиберы этого потока. Так как потоки фактически не работают одновременно, конфликтов при обращении к глобальной переменной (т.е. общей памяти) не возникает. Виртуальный параллелизм выполнения потоков обеспечивается постоянным переключением потоков по истечении кванта времени. Квант не измеряется в каких бы то ни было единицах времени, а выражается целым числом. Для каждого потока хранится текущее значение его кванта. Когда потоку выделяется квант процессорного времени, это значит, что его квант устанавливается в начальное значение. Оно зависит от операционной системы. Например, для Win2000 Professional начальное значение кванта равно 6, а для Win2000 Server - 36. Это значение можно изменить в ключе реестра HKLM Win32PrioritySeparation. Выбрав новый поток, операционная система переключает контекст. Эта операция заключается в сохранении параметров выполняемого потока (регистры процессора, указатели на стек ядра и пользовательский стек, указатель на адресное пространство, в котором выполняется поток и др.), и загрузке аналогичных параметров для другого потока, после чего начинается выполнение нового потока. Для определения процессорного времени, выделяемого для работы одного потока, поточная функция выводила в файл время, возвращаемое функцией clock(). В результате было установлено, что процессорное время, выделяемое одному потоку, составляет 94-96 мс.

Было также исследовано поведение более чем двух потоков на одноядерном процессоре. Результаты исследования показали, что первым запускается поток, который первым вызывается функцией ResumeThread. По истечении кванта времени будет работать один из всех созданных потоков, при этом порядок выбора потока определяется случайным образом. В частности, может несколько квантов подряд отработать один и тот же поток. Однако это справедливо только для потоков с одинаковыми приоритетами. В Windows реализована система вытесняющего планирования на основе приоритетов, в которой всегда выполняется поток с наибольшим приоритетом, готовый к выполнению. Однако это не значит, что если какая-либо задача имеет более высокий приоритет, то пока она не выполнится, никакие другие задачи выполняться не могут. Т.к. ОС Windows управляется событиями, то, как только очередь сообщений задачи с большим приоритетом будет пуста, управление получит поток, относящийся к менее приоритетной задаче.

В отличие от потоков, переключение которых организует операционная система, волокна переключает сам разработчик именно в те моменты, которые по его мнению наиболее подходят для переключения. В этом заключается преимущество волокон по сравнению с потоками. Однако, выполняться параллельно могут только волокна на разных потоках.

Исследование многопоточности волокон на двух ядрах

Потоками и фиберами управляет ядро WinNT, а Windows 9x не работает с более чем одним процессором. Проведем те же исследования, только на двух ядрах (схема теста аналогична). Для этого выполним следующие действия:

```
SetThreadAffinityMask(hThred,0x00000001);
SetThreadAffinityMask(hThred,0x00000002);
```

На рис.2 показана загрузка ЦП для двух ядер.

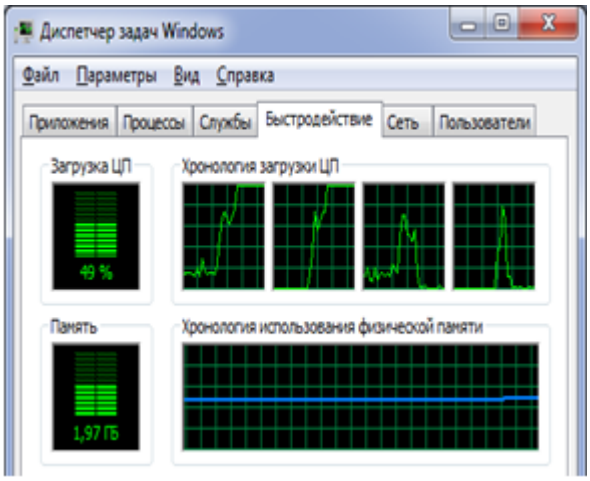

Рисунок 2 - Загрузка ЦП с 4 ядрами при выполнении программы на двух ядрах

В разработанном программном обеспечении также как и при проведении экспериментов на одноядерном процессоре, поточная функция преобразовывала поток в волокно и создавала новое волокно, а фиберная функция выводила в файл текущее процессорное время и наращивала глобальную переменную. Результаты работы программы подтвердили теоретические предположения, что потоки и, соответственно волокна, работают параллельно. Обращение к общей области памяти также не вызывает конфликтов при работе программы. Как показали результаты исследования программы на процессоре Intel Core 2 Duo, оба потока могут одновременно обращаться к одной ячейке памяти практически без задержек, что может быть полезным при реализации параллельных фрагментов программы. Это объясняется тем, что в современных многоядерных процессорах кэш второго уровня L2 является памятью раздельного пользования общим на два ядра, что значительно упростило обмен информацией между ядрами, снизив задержки, возникающие при работе обеих ядер с одним и тем же набором данных.

Заключение

Использование многопоточности может значительно ускорить работу приложения, особенно на многоядерном процессоре. Особую актуальность приобретает использование фиберов (волокон), т.к. переключение фиберов инициируется программистом в наиболее подходящий момент времени, в отличие от переключения потоков, которое инициирует операционная система. Установлено, что время, которое операционная система Windows 7 выделяет потоку, составляет 94-96мс. Исследовано также обращение из различных потоков к разделяемой общей памяти и выявлено, что при этом не возникает конфликтов на современных процессорах, что является важным при распараллеливании кода приложения. Дальнейшие исследования связаны с возможностью использования волокон для организации защиты программы от несанкционированного доступа. Такое использование волокон основано на том, что если фиберная функция будет выполнять идентификацию компьютера, на котором запускается приложение, взломать такую защиту будет достаточно сложно, т.к. взломщику достаточно трудно оценить момент, когда будет вызываться эта фиберная функция.

Список литературы

1. Интернет-ресурс - http://www.vbstreets.ru/VB/Articles/66058.aspx А. Скробов. Использование волокон ("фиберов") в VB для одновременного выполнения нескольких задач. – 2006.

2. Интернет ресурс - http://wm-help.net/books-online/book/59464.html. Дж. Рихтер. «Windows для профессионалов»

3. Рихтер. Дж. Windows via C/C++. Программирование на языке Visual C++ 2 / К. Назар, Дж. Рихтер.// Питер, Русская Редакция, 2009 г. – 896с.

4. Интернет ресурс - http://msdn.microsoft.com/en-us/library/hskdteyh%28v=vs.80%29.aspxPaul S. Heckbert. Survey of Texture Mapping / Paul S. Heckbert // IEEE Comput. Graph. and Applicat. – 1986. – 6, No 11. – P. 56-67.

Надійшла до редколегії 10.08.2013

О.Ю. ЧЕРЕДНИКОВА, О.Ю. ІВАНОВ, Є.В. КОЦОГУБ
Донецький національний технічний університет

ДОСЛІДЖЕННЯ БАГАТОПОТОЧНОСТІ З ВИКОРИСТАННЯМ ВОЛОКОН НА ОДНО - ТА БАГАТОЯДЕРНОМУ ПРОЦЕСОРАХ

В даній роботі розглядаються принципи організації багатопоточності з використанням волокон на одно та багатоядерному процесорах. Порівнюється організація багатопоточності на основі потоків та волокон. Досліджується звертання з різних потоків до загальної пам'яті. Здійснена спроба визначення процесорного часу, що виділяється кожному потоку операційною системою. Аргументуються переваги використання волокон для прискорення роботи програми на багатоядерному процесорі.

Ключові слова: потік, волокно, фібер, багатопоточність, багатоядерний процесор, процес, загальна пам'ять, процесорний час.

O.Yu. CHEREDNIKOVA, A.Yu. IVANOV, E.V. KOTSOGUB
Donetsk National Technical University

STUDY MULTITHREADING USING FIBERS AT SINGLE - AND MULTI-CORE PROCESSORS

In this work, the principles of organization of multithreading using threads and fiber to single - and multi-core processor are considered, their comparative characteristics are carried out. Using multithreading can greatly speed up the work of application, especially on multi-core processor. On single-core processor it is not possible to realize the true parallel work of threads. Each thread allocates some CPU time, after which there is a thread switching. It is established that the Windows 7 operating system allocates 94-96 msec for each thread. Moreover, if there are more than two threads with the same priority, it is impossible to reliably predict which of the threads to be executed. If the threads have different priorities, then run a higher priority thread. In this regard, of particular relevance gets fibers, because switching fibers is initiated by the programmer in the desired points in time, in contrast to switch threads that are switched by the operating system. However, on one thread can't run in parallel several fibers. Fibers can run in parallel only on different threads.

In the work we studied the possibility of calling from various functions of fiber and thread to the shared memory. As have shown results of program research on the processor Intel Core 2 Duo, both threads can simultaneously access a single memory cell almost without delay, which can be useful when implementing parallel fragments of the program. This is because in today's multi-core processors L2 cache is a memory of separate use and shared on two cores, thus considerably simplifies the exchange of information between the cores, reducing the delays in the work of both cores with the same set of data.

Further research is related to the use of fiber for the organization of protection against unauthorized access to information. Such use of fibers based on the fact that if fiber function will perform the identification of the computer running in place, to break this protection would be rather difficult, because the attacker is quite difficult to estimate the moment when this fiber function will be called.

Keywords: thread, fiber, multi-core processor, process, multithreading, shared memory and CPU time.