УДК 004.457

П. С. Василевич, **С. М. Егоров**, **С. И. Бобренок**, **Е. А. Загурских**

ООО «Параллелз»
ул. Октябрьская магистраль, *4*, *Новосибирск*, *630007*, *Россия*

pvasilevich@odin.com, *segorov@odin.com*, *sergey.bobrenok@gmail.com*
eugeny.zagurskih@yandex.ru

РЕШЕНИЕ ПРОБЛЕМЫ СИНХРОНИЗАЦИИ ПРОЦЕССОВ ПРИ ОБРАЩЕНИИ К РАЗДЕЛЯЕМОЙ ПАМЯТИ ПОД УПРАВЛЕНИЕМ BOOST [*]

Предлагается решение, направленное на устранение источника зависаний процессов в Windows-системах с разделяемой памятью под управлением Boost. Решение основывается на замене предоставляемых по умолчанию библиотекой Boost разделяемых объектов синхронизации межпроцессного взаимодействия на объекты сторонней разработки. Представлены результаты экспериментов, подтверждающие эффективность предлагаемого решения.

Ключевые слова: Boost, разделяемая память, межпроцессное взаимодействие, синхронизация.

Введение

Организация разделяемой памяти – самый быстрый и простой механизм межпроцессного взаимодействия. Однако за кажущейся простотой скрываются большие проблемы, присущие системам этого типа, решение которых требует значительных трудозатрат. Наиболее сложным звеном в таких системах является синхронизация обращений процессов (клиентов разделяемой памяти) к общеиспользуемым данным памяти.

При проведении работы по теме «Программный комплекс (платформа) для организации и поддержки облачного хостинга клиентских информационных ресурсов (ПК АОХ)» возникла необходимость в разработке синхронизатора обращений клиентов к ресурсам памяти, роль которого была отведена модулю блокировок Lock Manager. Функционирование этого модуля предусматривало применение логики библиотеки Boost для межпроцессного взаимодействия. Использование библиотеки Boost [1] в ее оригинальном виде приводило к частым сбоям в работе модуля блокировок, возникающих из-за незавершенности (далее – зависания [2]) какого-либо процесса. Анализ сбоев выявил фундаментальный недостаток в логике работы синхронизирующих объектов библиотеки Boost, что заставило отказаться от использования оригинальной ее версии и создать собственную со своим механизмом синхронизации процессов.

[*] Работа выполнена в ООО «Параллелз» в рамках проекта по договору № 02.G25.31.0054 от 12 февраля 2013 г. с Минобрнауки России.

[1] См.: https://ru.wikipedia.org/wiki/Boost (дата обращения 01.04.2015).

[2] Зависание – компьютерное явление, при котором одна или несколько программ либо вся операционная система перестают реагировать на действия пользователя, либо начинают без остановки выполнять одну и ту же (необязательно полезную или содержательную) операцию, не реагируя на сообщения от других программ.

Василевич П. С., *Егоров С. М.*, *Бобренок С. И.*, *Загурских Е. А.* Решение проблемы синхронизации процессов при обращении к разделяемой памяти под управлением Boost // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2015. Т. 13, вып. 2. С. 28–33.

В настоящей работе основное место отводится проведению эксперимента по устранению возникающих зависаний процессов в Windows-системах с разделяемой памятью под управлением Boost.

Взаимодействие процессов
в системах с разделяемой памятью

Межпроцессное взаимодействие организуется путем разделения участков виртуального адресного пространства и обмена данными через разделяемую память. Для устранения конфликтных ситуаций при одновременном обращении двух и более процессов к одному и тому же сегменту памяти используются семафоры, обеспечивающие взаимодействие с ресурсом памяти в отдельный момент времени только одному процессу. Обычно семафор включает следующие элементы: признак занятости; идентификатор последнего из процессов, работавшего с семафором; количество процессов в очереди на работу с семафором. Элемент признак занятости информирует клиентов (процессов) о состоянии семафора – закрыт он или открыт.

Процесс, получивший доступ к ресурсу памяти, закрывает семафор, блокируя тем самым доступ к памяти остальным процессам, и становится активным по отношению к ним. Если по какой-то причине активный процесс длительное время не завершается, остальные из очереди процессы приостанавливают свою работу до открытия семафора.

Невозврат семафора в исходное состояние приводит к конфликтным ситуациям, разрешение которых не всегда бывает успешным. Причинами возникновения таких ситуаций могут быть ошибки программирования, а также сигналы, внезапно прерывающие выполнение процесса. Если активный процесс неожиданно прекращает свое существование, в захватившей им памяти могут оставаться ненужные или искаженные данные, восстановление которых не представляется возможным из-за исчезновения процесса. Обеспечение надежности функционирования механизма синхронизации возлагается на операционную систему, использующую для этого специальные инструменты и технологии.

Разделяемая память
под управлением библиотеки Boost

Одним из наиболее распространенных способов решения задачи синхронизации процессов при их обращении к разделяемой памяти является использование библиотеки Boost. Базовый класс библиотеки Boost.Interprocess позволяет создавать в качестве объекта семафора различные примитивы синхронизации. Выбор конкретного объекта синхронизации для работы с разделяемой памятью определяется при задании шаблона объекта памяти, программный код на языке C++ которого представлен на рис. 1.

```
template  <class CharType
          ,class  MemoryAlgorithm
          ,template <class IndexConfig> class  IndexType>
class  basic_managed_shared_memory;

typedef  basic_managed_shared_memory
          <char
          ,rbtree_best_fit <mutex_family>
          ,iset_index>
managed_shared_memory;
```

Рис. 1. Программный код шаблона объекта памяти

Создаваемый по шаблону объект класса MemoryAlgorithm [3] содержит, кроме самого алгоритма выделения памяти, также и предлагаемые объекты синхронизации mutex_family для работы с памятью.

Организация разделяемой памяти под управлением Boost [4] предусматривает размещение механизма синхронизации непосредственно в самой памяти. Другими словами, по стандартному алгоритму выделения памяти в ней создается заголовок (структура) и внутрь этого заголовка неявно помещается объект синхронизации. При этом на платформе Windows используется стандартный объект синхронизации spin-mutex, принцип работы которого построен на атомарном доступе к специальной числовой переменной, находящейся внутри spin-mutex. По сути, логика работы объекта MemoryAlgorithm создает рекурсию управления доступом к метаинформации, что снижает надежность механизма синхронизации.

На рис. 2 показан пример обращения к разделяемой памяти двух выполняемых процессов – клиентов 1 и 2. Первым из клиентов, установившим значение spin_mutex в 1, является клиент 1. Клиенту 2 доступ к разделяемым данным заблокирован и будет открыт только тогда, когда клиент 1 переустановит значение spin_mutex в 0. Если по какой-то причине клиент 1 самостоятельно не сможет обнулить spin_mutex, доступ к разделяемым данным будет находиться в закрытом состоянии. Следует заметить, что управление разделяемой памятью при этом не будет информировано и не сможет открыть доступ к разделяемым данным, пока функционирует хотя бы один из клиентов. Это является фундаментальным недостатком в существующей логике Boost межпроцессной синхронизации, проявление которого может привести к непредсказуемым последствиям.

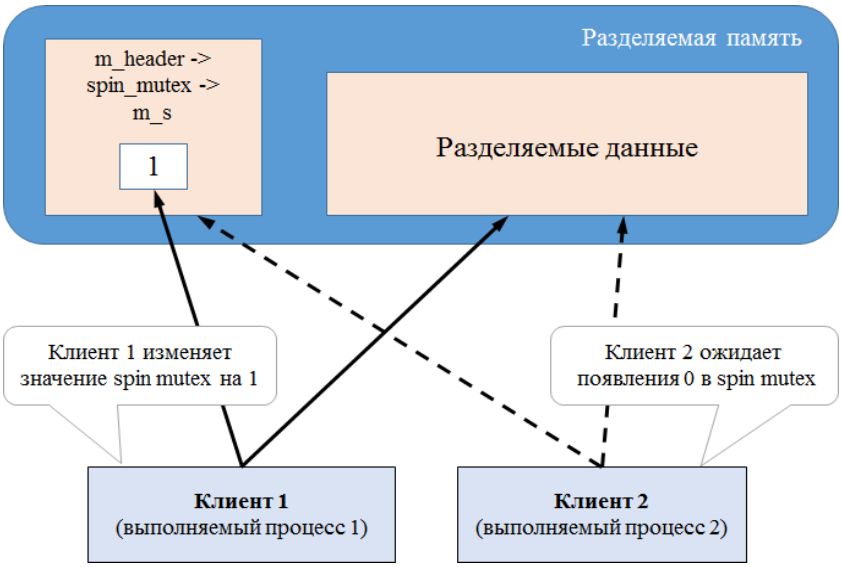

Рис. 2. Пример обращения клиентов к разделяемой памяти

Почему именно Boost применяется в Lock Manager

Поскольку модуль блокировок Lock Manager разрабатывается с учетом функционирования на многих операционных системах, возникла необходимость в абстрагировании от конкретной платформы и реализации на языке C++ кросс-платформенной логики по работе с разделяемой памятью, файловой системой, блокировками процессов. Использование в Lock

[3] См.: http://www.boost.org/doc/libs/1_55_0/doc/html/interprocess/managed_memory_segments.html#interprocess.managed_memory_segments.managed_memory_segment_features.synchronization (дата обращения 01.04.2015).

[4] См.: http://www.boost.org/doc/libs/1_53_0/doc/html/interprocess.html (дата обращения 01.04.2015).

Manager библиотеки Boost позволяет также абстрагироваться от различий в компиляторах, используемых на разных платформах (например, конструкции стандарта C++11 могут быть доступны на одних системах и закрыты на других).

Поиск решения проблемы

Использование в Boost в качестве хранилища метаинформации тех же самых разделяемых объектов таит в себе двойную неприятность. С одной стороны, это наличие случайных сбоев в системе, проявляющих себя как случайные зависания процессов, с другой – трудно диагностируемые ситуации. Следует заметить, что такая глубинная проблема изначально была и продолжает существовать в Boost, основательно камуфлируя истинный источник неприятностей.

Анализ возможных вариантов устранения данного недостатка показал, что наиболее эффективным решением является отказ от использования существующего механизма синхронизации в Boost и создание собственного. Boost предоставляет пользователям возможность заменять поставляемые объекты синхронизации на объекты сторонних реализаций.

В рамках работы по реализации модуля блокировок Lock Manager в Plesk Panel было решено провести эксперимент по устранению выявленного недостатка Boost с использованием механизма синхронизации в системе Windows, разработанного в ООО «Параллелз». Новизна решения заключалась в создании такого объекта, который по-прежнему может использоваться библиотекой Boost для синхронизации работы памяти, но который информировал бы клиентов о занятости памяти с помощью специфичных для операционной системы объектов ядра. Такие объекты размещаются вне пространства разделяемой памяти и поэтому могут быть освобождены операционной системой по завершении создавшего их процесса независимо от того, как будет завершен процесс: с ошибкой или корректно. Данное решение основано на создании своих объектов класса mutex, взаимодействующих с библиотекой Boost в соответствии с ее специфическим интерфейсом.

На рис. 3 показано обращение клиентов к разделяемым данным памяти в предлагаемой для проведения эксперимента реализации. Механизм синхронизации, размещенный вне области разделяемой памяти, исключает возможность создания нежелательного *разделяемого* доступа. Объект, по которому происходит синхронизация обращений, размещается вне объекта mutex, а значит, и вне разделяемой памяти. Блокировка таких объектов осуществляется на уровне ядра операционной системы за счет системных вызовов.

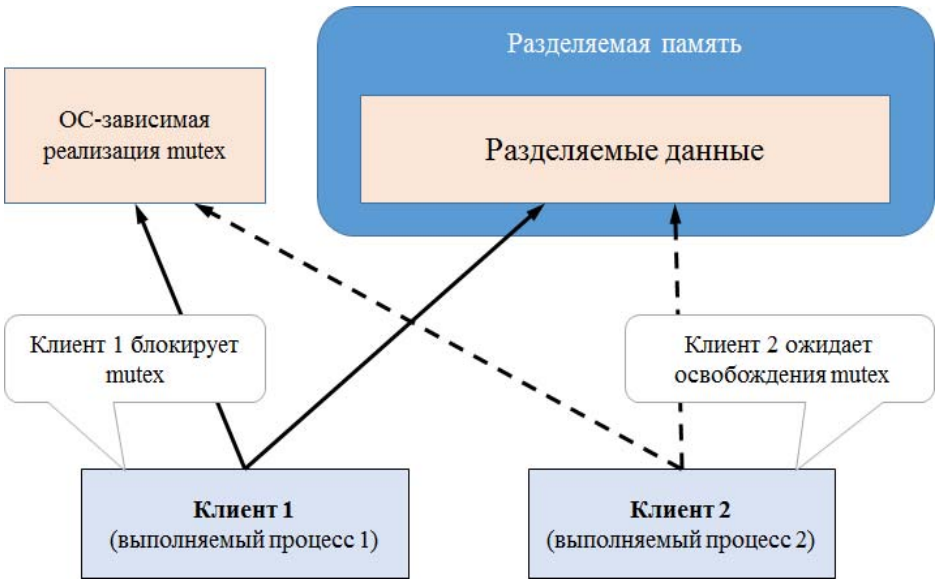

Рис. 3. Взаимодействие клиентов с разделяемыми данными в новой реализации

Результаты проведенного эксперимента

Для проверки эффективности предложенного решения было проведено экспериментальное тестирование на оборудовании с конфигурацией:

- процессор CPU Intel Core i7-960, 3.20 ГГц;
- оперативная память RAM 2 Гбайт;
- дисковая память HDD 500 Гбайт, 7200 об./мин;
- платформа Win32;
- версия Boost 1.53.

Для тестирования была написана специальная программа, способная инициировать одновременно несколько процессов, обращающихся к ресурсам разделяемой памяти. На фоне работающих процессов один из них в произвольный момент времени удалялся командой kill. При этом работа остальных процессов продолжалась в течение установленного (планового) времени. Для создания потенциально возможных конфликтных ситуаций каждый из процессов производил блокировку / разблокировку захваченного им ресурса памяти. Исследуемым объектом был модуль Lock Manager, представляющий собой интерпретатор PHP [5], управляемый тестовой программой.

Прохождение отдельного теста считалось успешным, если все процессы, за исключением удаленного, успевали завершиться за отведенное им время. В случае успешного прохождения теста он фиксировался в протоколируемом файле, после чего автоматически перезапускался и тестирование продолжалось. Если хотя бы один из процессов не завершался и при этом не отвечал на команды управляющей программы, прохождение теста считалось неуспешным и тестирование прекращалось с записью сообщения «fail» в протоколируемом файле. Продолжительность тестирования ограничивалась отведенным для него временем – 72 часа.

Результаты эксперимента приведены в таблице, где первая колонка показывает номера запусков тестовой программы, вторая – используемые объекты синхронизации в исследуемом объекте, третья – продолжительность тестирования и четвертая – причину прекращения тестирования.

Первые 10 прохождений тестовой программы показывают результаты работы интерпретатора PHP со старыми объектами синхронизации Boost, последние 3 прохождения – результаты с тестируемым решением.

Результаты работы теста с оригинальным и новым объектами синхронизации

№	Объекты синхронизации	Время работы программы	Причина останова программы
1	MemoryAlgorithm	7 с	fail
2	MemoryAlgorithm	3 с	fail
3	MemoryAlgorithm	11 с	fail
4	MemoryAlgorithm	8 с	fail
5	MemoryAlgorithm	14 с	fail
6	MemoryAlgorithm	4 с	fail
7	MemoryAlgorithm	9 с	fail
8	MemoryAlgorithm	6 с	fail
9	MemoryAlgorithm	11 с	fail
10	MemoryAlgorithm	7 с	fail
11	Win32InterprocessRecursiveMutex	72 ч	лимит времени
12	Win32InterprocessRecursiveMutex	72 ч	лимит времени
13	Win32InterprocessRecursiveMutex	72 ч	лимит времени

[5] См.: https://ru.wikipedia.org/wiki/PHP (дата обращения 01.04.2015).

Количество успешных прохождений теста на объектах Win32InterprocessRecursiveMutex за 72 часа составляло более 23 000, в то время как на используемых по умолчанию библиотекой Boost – не более 5.

Заключение

Возникающие зависания процессов в работе с разделяемой памятью приводят к непредсказуемым последствиям, вплоть до потери пользовательских данных, и могут иметь место всюду, где применяется логика синхронизации процессов Boost. Решение по устранению этой проблемы в системе Windows, предложенное ООО «Параллелз» в рамках работы по созданию модуля блокировок Lock Manager и основанное на создании собственной библиотеки с применением нового механизма синхронизации процессов, является простым в реализации и при этом вписывается в парадигму Boost. Результаты проведенного эксперимента показывают высокую эффективность предложенного решения.

Материал поступил в редколлегию 03.06.2015

P. S. Vasilevich, **S. M. Egorov**, **S. I. Bobrenok**, **E. A. Zagurskikh**

OOO «Parallels»
4 Oktyabrskaya magistral Str., *Novosibirsk*, *630007*, *Russian Federation*

pvasilevich@odin.com, *segorov@odin.com*, *sergey.bobrenok@gmail.com*
eugeny.zagurskih@yandex.ru

RESOLVING THE PROCESS SYNCHRONIZATION PROBLEM WHEN WORKING WITH SHARED MEMORY MANAGED BY BOOST

The article deals with cross-platform solution for eliminating the hangs of processes in Windows systems with using a shared memory. The decision is based on creating new synchronization mechanism, instead of that one which is used in Boost libraries by default. Experimental results confirm the efficiency of the proposed solution.

Keywords: Boost, shared memory, interprocess communication, synchronization.