Досліджено вплив технології програмної транзакційної пам'яті на продуктивність розподілених сховищ даних в оперативній пам'яті. Розглянуто локальний та розподілений алгоритми транзакційної памяті, проведені дослідження простих та складних запитів в умовах триразової реплікації та за її відсутності. Визначено особливості роботи програмної транзакційної пам'яті в умовах постійно навантаженого розподіленого сховища даних в оперативній пам'яті. Запропоновано рекомендації стосовно підвищення продуктивності сховища з використанням транзакційної пам'яті

Ключові слова: розподілене сховище даних, сховище в оперативній пам'яті, транзакційна пам'ять, програмна транзакційна пам'ять

Исследовано влияние технологии программной транзакционной памяти на производительность распределенных хранилищ данных в оперативной памяти. Рассмотрены локальный и распределенный алгоритмы транзакционной памяти, проведены исследования простых и сложных запросов в условиях трехкратной репликации данных и при ее отсутствии. Определены особенности работы программной транзакционной памяти в условиях постоянно нагруженного распределенного хранилища данных в оперативной памяти. Предложены рекомендации по повышению производительности хранилища с использованием транзакционной памяти

Ключевые слова: распределенное хранилище данных, хранилище в оперативной памяти, транзакционная память, программная транзакционная память

УДК 004.652.5
DOI: 10.15587/1729-4061.2015.48241

ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ХРАНИЛИЩ ДАННЫХ В ОПЕРАТИВНОЙ ПАМЯТИ ПОСРЕДСТВОМ ИСПОЛЬЗОВАНИЯ ПРОГРАММНОЙ ТРАНЗАКЦИОННОЙ ПАМЯТИ

А. А. Подрубайло
Аспирант, ассистент
Кафедра вычислительной техники
Национальный технический университет Украины «Киевский политехнический институт»
пр. Победы, 37, г. Киев, Украина, 03056
E-mail: sxl_sas@ukr.net

1. Введение

Распределенные хранилища данных в оперативной памяти (In-memory data grid, IMDG) на сегодняшний день достаточно широко распространены. Эта технология призвана уменьшить время доступа к данным путем хранения их на быстродействующем носителе – оперативном запоминающем устройстве (ОЗУ) [1]. С удешевлением модулей ОЗУ и одновременным ростом их ёмкости растет и популярность IMDG. Однако использование транзакций в существующих хранилищах такого рода существенно снижает производительность этих систем. Падение производительности обусловлено использованием блокировок для реализации транзакций в современных IMDG. Их применение накладывает существенные ограничения на многопоточную работу с данными.

Альтернативой блокировкам в этом случае может выступить использование транзакционной памяти (transactional memory, TM). Так как к IMDG выдвигаются требования по устойчивой работе в гетерогенных распределенных системах, целесообразнее рассматривать программную транзакционную память (software transactional memory, STM), т. к. прочие подходы к реализации TM накладывают существенные ограничения на аппаратную составляющую системы.

Концепция транзакционной памяти близка к принципам ACID (atomicity, consistency, isolation, durability), лежащим в основе большинства современных баз данных. Тем не менее, TM не применяется в современных реализациях распределенных хранилищ в оперативной памяти. Напротив, трудности с использованием транзакций привели к появлению новой идеологии BASE (basically available, soft-state, eventually consistent) [2], предлагающей считать временную недоступность данных и частичную неконсистентность информации в распределенной системе нормальным поведением такого класса систем.

2. Анализ литературных данных и постановка проблемы

Аппаратная версия транзакционной памяти была предложена Лометом в 1977 [3], реализация ее была впервые описана Херлихи и Моссом в 1993 [4]. TM следует концепции оптимистичного параллелизма, при котором любая транзакция выполняется в предположении, что у нее не будет конфликтов с другими тран-

закциями. Если две транзакции конфликтуют, поскольку одна из них модифицирует некоторую область памяти, прочитанную или модифицированную другой транзакцией, то система TM аварийно завершает одну из этих транзакций путем устранения произведенных изменений (отката) [5].

При параллельном программировании приходится сталкиваться со многими трудностями, но одной из наиболее серьезных проблем при написании корректного кода является координация доступа к данным, совместно используемым в нескольких потоках управления. Последствиями попыток обеспечить взаимное исключение со слишком слабой или слишком сильной синхронизацией являются конфликты при доступе к данным (data race), синхронизационные тупики (deadlock) и низкая способность к масштабируемости. TM является более простой альтернативой взаимного исключения, обеспечивая корректную синхронизацию без участия программиста [6].

TM обеспечивает механизм легковесных транзакций для потоков управления, выполняемых в общем адресном пространстве, гарантирует атомарность и изолированность параллельно выполняемых задач [4, 5].

Атомарность означает, что изменения, производимые транзакцией, не могут быть выполнены частично. При обнаружении конфликта одна из транзакций будет отменена, при этом все изменяемые ею данные будут возвращены в исходное состояние.

Изолированность гарантирует, что параллельно выполняемые задачи не влияют на результат транзакции, так что транзакция производит один и тот же результат, как если бы не выполнялась никакая другая задача.

Программная транзакционная память была впервые предложена в Шавитом и Туиту в 1995 [7], однако первая ее реализация требовала заранее определить адреса переменных, к которым будет осуществлен доступ, что не применимо в условиях IMDG. В [8] и [9] предложены динамические реализации STM, отличающиеся степенью зернистости. Крупнозернистая реализация [8] предполагает отслеживание изменений всего объекта, с которым работает транзакция. Мелкозернистый подход [9] заключается в отслеживании состояния отдельных полей используемого объекта, что дает возможность успешного выполнения транзакций, одновременно изменяющих различные поля в одном и том же объекте.

Традиционной областью применения транзакционной памяти считаются системы с общей памятью, но в [10] и [11] предложены распределенные модели STM, демонстрирующие применимость этой технологии в распределенных системах.

В современных IMDG не используется транзакционная память, однако в некоторых простейших системах кеширования (MemC3) реализован оптимистичный подход к изменению данных [12]. В [13] предложено вынести контроль над транзакциями на уровень промежуточного приложения (middleware) с использованием STM, однако такой подход имеет ряд существенных ограничений, как то низкая отказоустойчивость, ограниченная доступность и высокие накладные расходы на работу промежуточного слоя.

В то же время, исключение механизма блокировок и переход на программную транзакционную память непосредственно внутри хранилища потенциально способно исключить накладные расходы (т. к. отсутствует промежуточный слой), повысив тем самым скорость обработки транзакций. Однако на сегодняшний день такая архитектура распределенных хранилищ данных не была изучена. Таким образом, исследование влияния программной транзакционной памяти на производительность хранилищ класса IMDG является актуальной задачей.

3. Цель и задачи исследования

Целью данной работы является исследование возможности повышения производительности распределенных хранилищ данных в оперативной памяти путем использования программной транзакционной памяти.

Для достижения поставленной цели решались следующие задачи:

– определение производительности распределенного хранилища данных в оперативной памяти при использовании глобальной транзакционной памяти;

– определение производительности IMDG при использовании локальной транзакционной памяти;

– формулировка рекомендаций по повышению производительности распределенного хранилища в оперативной памяти путем использования STM.

4. Материалы и методы исследований производительности распределенного хранилища данных при использовании программной транзакционной памяти

4. 1. Аппаратная конфигурация исследуемой системы

Тестовый кластер состоит из четырех серверов с идентичной аппаратной конфигурацией: процессор Intel Xeon E5620 (8 ядер на частоте 2.4 ГГц), 128 Гб оперативной памяти с коррекцией ошибок. Узлы кластера соединены в одну локальную сеть Gigabit Ethernet.

4. 2. Конфигурация и наполнение тестового IMDG

Для исследований было выбрано хранилище данных в оперативной памяти, содержащее 16 узлов данных (по четыре на каждом сервере) и 4 узла-локатора, перенаправляющих клиентские запросы на конкретный узел данных. Клиенты могут обращаться к произвольному узлу-локатору, при этом их запрос может перенаправляться на свободный узел с требуемыми данными в пределах кластера. Конфигурация системы представлена на рис. 1.

В качестве наполнения была взята программа телепередач, состоящая из 15 000 провайдеров, 10 000 каналов, 100 000 программ, 2 000 000 расписаний и 1 000 000 контентов (рис. 2). Общий объем данных – 140 гигабайт.

В экспериментах использовались степени избыточности от 1 до 3 (т. е. для одной записи существовало максимум 2 копии на других узлах хранилища).

Для контроля влияния STM использовалось идентичное по конфигурации и наполнению распределенное хранилище, использующее блокировки.

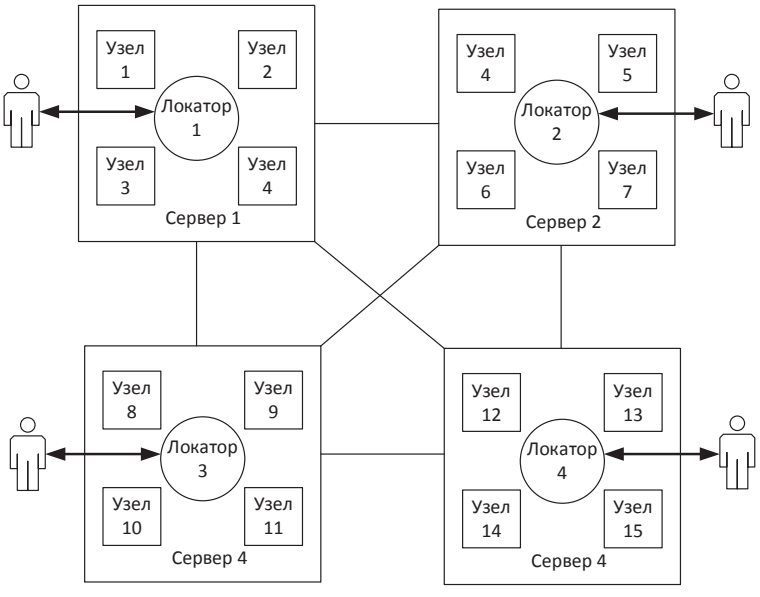

Рис. 1. Конфигурация исследуемой системы хранения данных в оперативной памяти

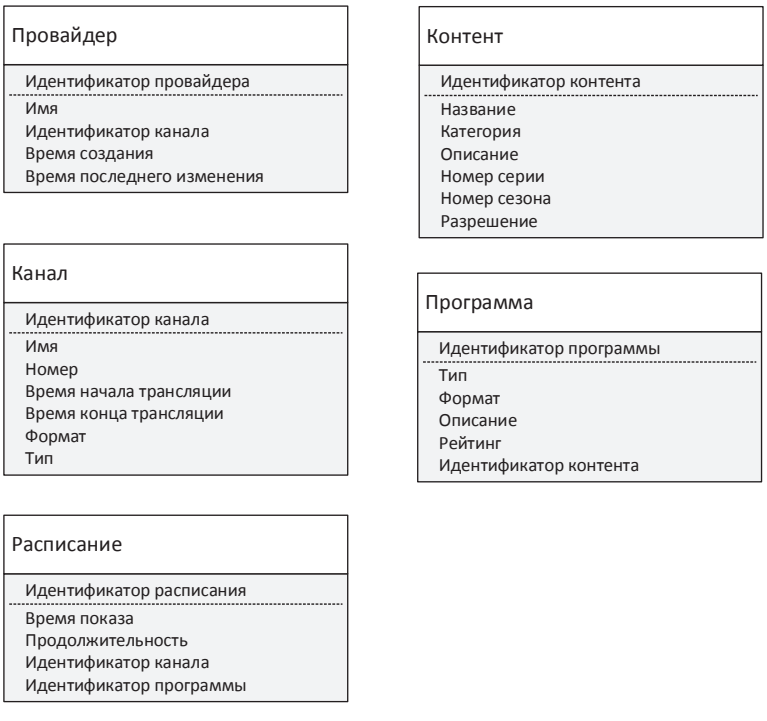

Рис. 2. Структура хранимых данных

4. 3. Используемые методы работы хранилища данных с программной транзакционной памятью

В рамках настоящего исследования использовалось несколько методов оптимистического контроля конкурентности:

1. Локальный. Заключается в использовании версии локального объекта в качестве метаданных. При любом изменении объекта транзакция инкрементирует его версию. В случае, если за время работы транзакции версия объекта изменилась, транзакция отменяется и не вносит каких-либо изменений.

Данный метод применим к распределенным хранилищам только в двух случаях: если хранимые данные не реплицированы либо если в хранилище допустима консистентность в конечном счете. В первом случае в IMDG хранится только один экземпляр каждого объекта, и сбой в хранящем его узле приведет к потере данных. Во втором случае одновременное изменение хранимых на разных узлах копий объекта приведет к неконсистентности хранилища и потребует в дальнейшем сложных алгоритмов арбитража для выяснения, какую копию данных считать верной.

В данном исследовании локальная транзакционная память использовалась при отсутствии репликации с асинхронным сохранением данных на постоянный носитель.

2. Глобальный. Этот метод оптимизирован для работы в распределенных хранилищах с репликацией данных. В архитектуру хранилища добавлен менеджер транзакций, повторяющий транзакцию для всех реплик конкретного объекта. Если хотя бы для одной копии транзакция не завершилась успешно – вся транзакция считается незавершенной и откатывается к предыдущему состоянию.

4. 4. Методика исследования влияния программной транзакционной памяти на производительность хранилища данных

Эксперименты проводились на заполненном на 70 % хранилище при фиксированном количестве запросов в секунду (2000). Варьировалось соотношение запросов чтения/записи: 10 %, 50 %, 70 %, 100 %. Здесь 10 % означает, что из 2000 запросов в секунду 200 были запросами на чтение информации и 1800 – на запись. Запросы равномерно распределялись по всем четырем серверам.

Использовалось три типа запросов чтения:

1) запрос контента по его идентификатору. Простейший запрос, выполняющий поиск и чтение произвольной реплики объекта по его основному ключу;

2) запрос канала и его расписаний за последние три часа по идентификатору канала. Составной запрос, вначале выполняющий поиск канала по идентификатору, затем, если канал найден, при помощи индексного запроса к расписаниям читающий все связанные с каналом расписания, активные в заданное время;

3) формирование 24-часовой выжимки расписания телепередач по 10 каналам. Сложный запрос, выполняющий поиск каналов и связанных с ними провайдеров, расписаний, программ и контентов посредством индексных запросов.

Все операции записи были однотипными, а именно: запись конкретного объекта по его идентификатору.

В рамках одного эксперимента использовался только один тип запросов чтения, для сравнительной оцен-

ки результатов каждый эксперимент повторялся в трех конфигурациях хранилища: с блокировками, с локальной и глобальной программной транзакционной памятью.

Данные для запросов выбирались случайным образом из заранее сформированного множества, содержащего 10 % всех идентификаторов хранилища.

Использовались конфигурации хранилища без избыточности и с двумя резервными копиями объекта. При этом, т.к. использование только локальной STM ведет к нарушению консистентности в хранилище с резервными копиями, эксперименты в хранилище с избыточностью проводились только в двух конфигурациях: с блокировками и с глобальной STM. В хранилище без избыточности с целью обеспечения отказоустойчивости использовалась асинхронное сохранение данных на долговременное запоминающее устройство (жесткий диск).

В качестве показателя производительности принято среднее время выполнения транзакции, которое определялось в результате часовой непрерывной работы хранилища под нагрузкой заранее оговоренного рода. Перед началом каждого испытания проводился 10-минутный разогрев, во время которого показатели производительности не снимались.

5. Результаты исследования производительности IMDG при использовании программной транзакционной памяти

Результаты экспериментов при отсутствии репликации хранимых данных показаны в табл. 1–3.

Таблица 1

Среднее время выполнения запроса на поиск контента по ключу в IMDG с различными механизмами конкурентности (репликация отсутствует), мс

Доля запросов чтения	10 %	50 %	70 %	100 %
Блокировки	27	30	26	25
Локальный STM	27	26	26	26
Глобальный STM	50	44	43	43

Таблица 2

Среднее время выполнения запроса на поиск канала и связанных расписаний за три последних часа в IMDG с различными механизмами конкурентности (репликация отсутствует), мс

Доля запросов чтения	10 %	50 %	70 %	100 %
Блокировки	2036	2341	2538	532
Локальный STM	1765	1472	1343	543
Глобальный STM	1989	1754	1632	622

Таблица 3

Среднее время выполнения запроса на поиск расписания телепередач 10 каналов за последние сутки в IMDG с различными механизмами конкурентности (репликация отсутствует), мс

Доля запросов чтения	10 %	50 %	70 %	100 %
Блокировки	3048	3745	4874	1000
Локальный STM	3576	3654	3484	1020
Глобальный STM	3987	4132	4345	1200

В табл. 4–6 показаны результаты выполнения аналогичных запросов при наличии двух дополнительных копий каждого объекта.

Таблица 4

Среднее время выполнения запроса на поиск контента по ключу в IMDG с различными механизмами конкурентности и репликацией, мс

Доля запросов чтения	10 %	50 %	70 %	100 %
Блокировки	26	30	25	23
Глобальный STM	87	90	78	74

Таблица 5

Среднее время выполнения запроса на поиск канала и связанных расписаний за три последних часа в IMDG с различными механизмами конкурентности и репликацией, мс

Доля запросов чтения	10 %	50 %	70 %	100 %
Блокировки	2354	2576	2604	487
Глобальный STM	1907	1740	1604	634

Таблица 6

Среднее время выполнения запроса на поиск расписания телепередач 10 каналов за последние сутки в IMDG с различными механизмами конкурентности и репликацией, мс

Доля запросов чтения	10 %	50 %	70 %	100 %
Блокировки	3508	4109	4536	951
Глобальный STM	3947	4070	3989	1608

6. Обсуждение результатов исследования производительности IMDG с применением технологии транзакционной памяти

Из результатов эксперимента видно, что применение транзакционной памяти при отсутствии операций записи не приводит к повышению производительности. Это объясняется тем, что конкурентное чтение не приводит к блокировкам, а, следовательно, дополнительные проверки метаданных в этом случае избыточны и лишь уменьшают производительность.

Также производительность хранилища не увеличивается при использовании только простых запросов типа «чтение по ключу», что объясняется простотой этой операции.

Однако в сложных запросах, требующих чтения индексов, ситуация меняется. При 10 % доле запросов чтения короткие транзакции выигрывают у блокировок тем, что успевают прочесть информацию из индекса непосредственно во время записи в индекс, в то время как запросы с блокировкой ждут своей очереди. Как видно из табл. 2 и 5, при наличии одновременных операций записи и чтения, поиск канала и связанных расписаний за три последних часа с применением локальной STM приводит к приросту производительности от 13,3 %–18,9 % (при 10 % доле запросов чтения) до 38,4 %–47 % (при 70 % запросов чтения). Причиной такого прироста является изменившийся конкурентный подход к работе с общими данными. При исполь-

зовании блокировок запись произвольного объекта вызовет перестройку сегмента индекса, на время которой данный сегмент будет заблокирован (недоступен для чтения). В то же время при использовании STM изменяемый сегмент будет доступен для чтения непосредственно до момента его перезаписи, что позволяет достаточно коротким транзакциям выполниться быстрее, чем их аналогам с блокировками.

Из табл. 3 и 6 видно, что применение транзакционной памяти для сложного запроса (расписания телепередач 10 каналов за последние сутки) при 10 % запросов чтения ведет к увеличению времени обработки запроса на 12,5 %–17,3 %. Причиной этого является увеличившаяся продолжительность транзакции, во время которой происходит чтение нескольких индексов. При этом произвольное изменение индекса, происходящее при записи данных, ведет к откату транзакции и дальнейшему ее повтору. В ситуации, когда количество операций чтения и записи уравнивается, индексы обновляются не так часто и транзакционная память также получает преимущество перед механизмом блокировок (0,9 %–2,5 %). При 70 % доле запросов чтения это преимущество более заметно и составляет 12–28,5 %.

Метод глобальной транзакционной памяти на 10–15 % проигрывает локальному аналогу, однако последний применим лишь при отсутствии избыточности, необходимой во многих современных приложениях для обеспечения надежности и отказоустойчивости.

На основании проведенных исследований можно сформулировать следующие рекомендации по повышению производительности IMDG с использованием программной транзакционной памяти:

1. Программную транзакционную память следует использовать в приложениях, совмещающих фоновую запись данных в хранилище и частые операции чтения. Примером могут послужить хранилища данных для бизнес-анализа, базы данных ТВ-гида и др.

2. При отсутствии репликации в хранилище следует использовать локальный алгоритм STM, в хранилищах с избыточностью – глобальный.

3. Длинные операции чтения следует разбивать на составляющие, чтобы транзакции были максимально короткими.

4. В хранилищах, где одновременного чтения и записи данных не происходит или операции записи данных составляют менее 5 % от общего числа операций (например, библиотечные каталоги, базы расписания занятий, архивные системы и т. д.), применение программной транзакционной памяти не оправдано и ведет к падению производительности.

7. Выводы

В данной работе проведены исследования производительности IMDG с программной транзакционной памятью, предложена методика повышения производительности распределенного хранилища в оперативной памяти посредством использования STM.

Замена механизма управления конкурентностью с блокировок на программную транзакционную память и уменьшение размера транзакций позволяет добиться прироста производительности распределенных хранилищ данных в оперативной памяти на 13–48 %.

Предложенная методика применима для хранилищ, совмещающих фоновую запись данных в хранилище и частые операции чтения. В хранилищах, где для чтения данных применяется исключительно выборка по ключу, предложенная методика не применима. Также программную транзакционную память нецелесообразно применять в тех IMDG, для которых операции записи составляют менее 5 % всех запросов.

Литература

1. Williams, J. W. Bridging high velocity and high volume industrial big data through distributed in-memory storage & analytics [Text] / J. W. Williams, K. S. Aggour, J. Interrante, J. McHugh, E. Pool // 2014 IEEE International Conference on Big Data (Big Data), 2014. – P. 932–941. doi: 10.1109/bigdata.2014.7004325
2. Pritchett, D. BASE: an ACID alternative [Text] / D. Pritchett // Queue. – 2008. – Vol. 6, Issue 3. – P. 48–55. doi: 10.1145/1394127.1394128
3. Lomet, D. B. Process structuring, synchronization, and recovery using atomic actions [Text] / D. B. Lomet // Proceedings of the ACM Conference on Language Design for Reliable Software (Raleigh, NC), 1977. – P. 128–137.
4. Herlihy, M. Transactional memory: Architectural support for lock-free data structures [Text] / M. Herlihy, J. E. B. Moss // Proceedings of the 20th Annual International Symposium on Computer Architecture, 1993. – P. 289–300. doi: 10.1109/isca.1993.698569
5. Larus, J. Transactional Memory [Text] / J. Larus, C. Kozyrakis // Communications of the ACM. – 2008. – Vol. 51, Issue 7. – P. 80. doi: 10.1145/1364782.1364800
6. Grossman, D. The transactional memory/garbage collection analogy [Text] / D. Grossman // Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (Montreal, Canada), 2007. – P. 695–706.
7. Shavit, N. Software transactional memory [Text] / N. Shavit, D. Touitou // Proceedings of the 14th ACM Symposium on Principles of Distributed Computing (Ottawa, Canada), 1995. – P. 204–213.
8. Herlihy, M. Software transactional memory for dynamic-sized data structures [Text] / M. Herlihy, V. Luchangco, M. Moir, W. N. Scherer III // In Proceedings of the twenty-second annual symposium on Principles of distributed computing – PODC '03, 2003. – P. 92–101. doi: 10.1145/872035.872048
9. Fraser, K. Concurrent programming without locks [Text] / K. Fraser, T. Harris // ACM Transactions on Computer Systems. – 2007. – Vol. 25, Issue 2. – P. 5–51. doi: 10.1145/1233307.1233309

10. Carvalho, N. A generic framework for replicated software transactional memories [Text] / N. Carvalho, P. Romano // 2011 IEEE 10th International Symposium on Network Computing and Applications, 2011. – P. 271–274. doi: 10.1109/nca.2011.45
11. Saad, M. HyFlow: a high performance distributed software transactional memory framework [Text] / M. Saad, B. Ravindran // Proceedings of the 20th international symposium on High performance distributed computing - HPDC '11, 2011. – P. 265–266. doi: 10.1145/1996130.1996167
12. Zhang, H. In-memory big data management and processing: A survey [Text] / H. Zhang, G. Chen, B. C. Ooi, K. L. Tan, M. Zhang // EEE Transactions on Knowledge and Data Engineering. – 2015. – Vol. 27, Issue 7. – P. 1920–1948. doi: 10.1109/tkde.2015.2427795
13. Fernandes, S. Strongly Consistent Transactions for Enterprise Applications. Using Software Transactional Memory to Improve Consistency and Performance of Read-Dominated Workloads [Text] PhD dissertation / S. Fernandes. – Lisbon, 2014 – 208 p.

Отримано формальний, математичний опис узагальненого процесу управління конфігурацією безвідносно до об'єкта його застосування для сфери управління проектами. Була введена класифікація дій цього процесу, формально описаний склад його варіантів реалізації, виділені його керуючі параметри. Також була введена класифікація змін контрольованого об'єкта

Ключові слова: конфігурація, управління конфігурацією, проект, управління проектами, процес, оптимізація, зміна, формалізація

Получено формальное, математическое описание обобщенного процесса управления конфигурацией безотносительно к объекту его приложения для сферы управления проектами. Была введена классификация действий этого процесса, формально описан состав его вариантов реализации, выделены его управляющие параметры. Также была введена классификация изменений контролируемого объекта

Ключевые слова: конфигурация, управление конфигурацией, проект, управление проектами, процесс, оптимизация, изменение, формализация

УДК 658.511
DOI: 10.15587/1729-4061.2015.47292

РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ ОБОБЩЕННОГО ПРОЦЕССА УПРАВЛЕНИЯ КОНФИГУРАЦИЕЙ В УПРАВЛЕНИИ ПРОЕКТАМИ

С. И. Рудницкий
Аспирант
Кафедра бизнес-администрирования и управления проектами
Университет экономики и права «КРОК»
ул. Лагерная, 30-32, г. Киев, Украина, 03113
E-mail: sergey.rudnitskiy@gmail.com

1. Введение

Высокая изменчивость окружения современных проектов, а также их сложность и междисциплинарность поднимают проблему эффективной поддержки согласованности таких проектов. В сфере управления проектами (УП) эту задачу решают в рамках процесса общего управления конфигурацией (УК). При этом под согласованностью понимают такое состояние проекта, когда он, как создающая продукт система, имеет те и только те элементы, которые способствуют созданию продукта проекта. Эффективная поддержка согласованности проекта означает, прежде всего, возможность оптимизации процесса общего УК, который условно можно представить в виде двух взаимосвязанных и неразрывных частей: процесса УК проекта и процесса УК продукта [1–3]. Поэтому оптимизация процесса УК проекта затрагивает и процесс УК продукта. Можно утверждать, что процесс УК направленный на продукт исследован и описан достаточно глубоко для его эффективного осуществления, а процесс УК проекта исследован только на концептуальном уровне, что не дает возможности его эффективной реализации. Общность процессов УК этих двух объектов позволила составить концептуальную модель обобщенного процесса УК безотносительно к объекту его приложения [4]. Наличие такой модели позволяет описывать оба эти процесса на основе единой концептуальной базы. Следующим этапом в оптимизации процесса общего УК является формальное математическое описание обобщенного процесса УК объекта.

2. Анализ литературных данных и постановка проблемы

Первым шагом такой формализации является математическая модель абстрактного объекта управле-