УДК 004.25.000

ЭФЕКТИВНОСТЬ ARC И GARBAGE COLLECTOR: ИССЛЕДОВАНИЕ МЕХАНИЗМОВ УПРАВЛЕНИЯ ПАМЯТЬЮ

Половой А.А.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Научный руководитель: Романюк М. В. – ассистент кафедры информатики

Аннотация. В данной статье рассматриваются механизмы управления памятью, используемые в современных языках программирования, с акцентом на автоматическое управление памятью с помощью *ARC* (*Automatic Reference Counting*) и сборщика мусора (*Garbage Collector*). Анализируются принципы работы этих технологий, их преимущества и недостатки, а также влияние на производительность приложений. Основное внимание уделяется сценариям использования и оптимизации ресурсов, а также перспективам развития методов управления памятью.

Ключевые слова: *ARC, Garbage Collector*, управление памятью, производительность, программирование, автоматическое управление памятью.

Введение. С управлением памятью связано множество проблем, особенно в контексте разработки высокопроизводительных приложений. *ARC* и *Garbage Collector* представляют собой две основные технологии автоматического управления памятью, каждая из которых имеет свои особенности. Важно понимать, как эти механизмы влияют на производительность приложений и как выбрать подходящий метод в зависимости от требований проекта [1].

Основная часть. *Automatic Reference Counting* (*ARC*) является механизмом управления памятью, который используется в языках, таких как *Swift* и *Objective-C*. *ARC* отслеживает количество ссылок на объекты и освобождает память, когда ссылки на объект становятся равными нулю. Этот подход позволяет избежать утечек памяти и значительно упрощает разработку, так как программисту не нужно вручную управлять памятью [2].

Преимущества *ARC*:

– простота использования: разработчики не требуют глубоких знаний о механизмах управления памятью;

– высокая производительность в средах с низким уровнем параллелизма.

Недостатки ARC:

– возможные циклические ссылки, которые требуют дополнительных усилий для их устранения;

– потенциальные накладные расходы при частом увеличении и уменьшении счетчиков ссылок [2].

Сборщик мусора (*Garbage Collector, GC*) используется в языках, таких как *Java* и *C#*. *GC* автоматически управляет памятью, периодически проверяя, какие объекты больше не используются, и освобождая их. Существует несколько алгоритмов сборки мусора, таких как маркировка и сборка, а также алгоритмы поколений [3].

Преимущества *GC*:

– эффективно управляет памятью в многопоточных приложениях;

– автоматически освобождает память, что упрощает разработку.

Недостатки *GC*:

– непредсказуемое время пауз, когда приложение может временно приостановиться для сборки мусора;

– высокие накладные расходы в случае частых операций сборки.

При выборе между *ARC* и *GC* разработчики должны учитывать особенности проекта.

ARC может быть более предпочтительным для приложений с высоким уровнем производительности и низким параллелизмом, тогда как *GC* будет полезен в сценариях, где необходима гибкость и простота [4].

С учетом современных тенденций в разработке программного обеспечения, важным направлением является улучшение существующих механизмов управления памятью. Разработка адаптивных алгоритмов, которые могут динамически переключаться между *ARC* и *GC* в зависимости от нагрузки, может значительно повысить эффективность управления памятью.

Заключение. *ARC* и *Garbage Collector* представляют собой мощные инструменты для автоматического управления памятью, и каждый из них имеет свои уникальные преимущества и недостатки. Понимание механизмов работы этих технологий позволяет разработчикам более эффективно управлять ресурсами приложений, обеспечивая высокую производительность и стабильность. Перспективы развития в этой области обещают новые решения, способные улучшить управление памятью в будущем.

Список литературы

1. *Jones, R., Hosking, A., Moss, E. The Garbage Collection Handbook: The Art of Automatic Memory Management. – CRC Press, 2023. – 432 p. – ISBN 978-1032264832.*

2. *Wu, Q. et al. Toward Understanding Bugs in Swift Programming Language // 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS). – IEEE, 2023. – Vol. 23. – Pp. 117–127.*

3. *Van Putten, M., Kennedy, S. Java Memory Management: A Comprehensive Guide to Garbage Collection and JVM Tuning. – Packt Publishing Ltd, 2022. – 322 p. – ISBN 978-1803237044.*

4. *Vayadande, K. et al. A Novel Automatic Garbage Collector Tool for C/C++ Programs // 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES). – IEEE, 2023. – Vol. 1. – Pp. 657–662.*

UDC 004.25.000

ARC AND GARBAGE COLLECTOR EFFICIENCY: A STUDY OF MEMORY MANAGEMENT MECHANISMS

Polovoy A.A.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Romaniuk M.V. – assistant of the Department of Informatics

Annotation. This paper discusses memory management mechanisms used in modern programming languages, focusing on automatic memory management using ARC (Automatic Reference Counting) and Garbage Collector. The principles of these technologies, their advantages and disadvantages, as well as their impact on application performance are analyzed. The main attention is paid to scenarios of resource usage and optimization, as well as the prospects for the development of memory management methods.

Keywords: ARC, Garbage Collector, memory management, performance, programming, automatic memory management.