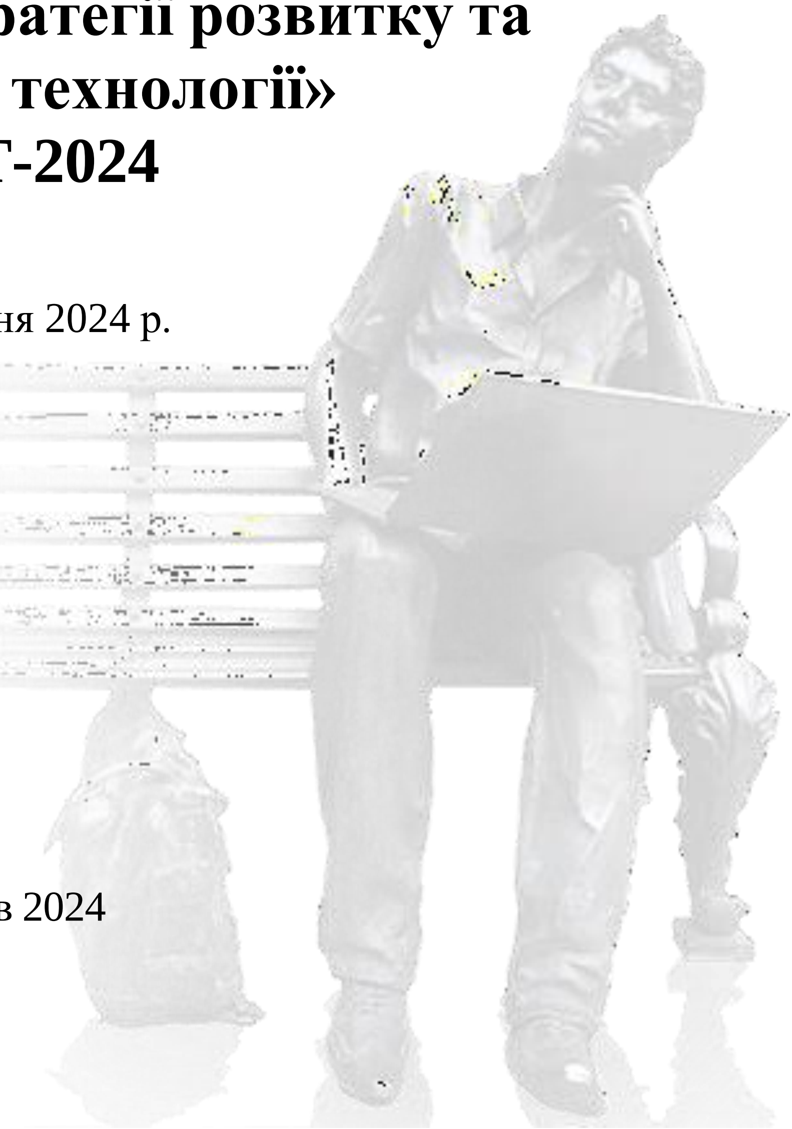
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ
VI ФОРУМУ
«Автоматизація, електроніка та робототехніка. Стратегії розвитку та інноваційні технології»
AERT-2024

6 - 7 грудня 2024 р.



Харків 2024

АДАПТАЦІЯ JAVA-ЗАСТОСУНКІВ ПІД ГІБРИДНІ ПРОЦЕСОРНІ АРХІТЕКТУРИ

ст. викл каф. МТС, Чумак В.С., студент Михайлюков М.В.
Харківський національний університет радіоелектроніки, кафедра інформаційно-вимірювальних технологій,
e-mail: mykhailo.mykhailiukov@nure.ua

Abstract. The paper addresses the problem of adapting Java applications to hybrid processor architectures that combine CPU, GPU and NPU. It describes the specific features of working with hybrid chips, the challenges developers face, and the prospects of using such architectures in modern software solutions. Significant attention is given to analyzing tools that help optimize Java code for interaction with GPUs and NPUs, particularly Aparapi, TensorFlow Java API, ONNX Runtime, Project Panama and GraalVM. The study emphasizes the efficient distribution of tasks among computational components, which enables substantial performance improvements and reduces the energy consumption of software.

Key words. Java, CPU, GPU, NPU, performance.

Introduction. Modern computational systems increasingly rely on hybrid processor architectures that combine a central processing unit (CPU), a graphics processing unit (GPU), and a neural processing unit (NPU). This combination enables efficient execution of a wide range of tasks, from general computation management to complex data processing and neural computations. The use of such architectures ensures high performance, energy efficiency, and the ability to optimally distribute workloads among different types of computational units.

Programming for hybrid architectures requires a specialized approach, particularly for high-level languages such as Java. Thanks to its platform independence and extensive ecosystem, Java remains a popular language for software development. However, integrating Java applications with GPUs and NPUs requires code adaptation, the use of third-party libraries, and tools for performance optimization. This paper examines the key challenges of adapting Java applications to hybrid architectures and the tools that help efficiently leverage the resources of such systems.

Main body. Hybrid processor architectures are designed to perform high-performance and energy-efficient computations. CPUs execute general computational tasks, GPUs provide high-speed data processing for parallelizable workloads, and NPUs excel in executing machine learning algorithms. These architectures enable complex operations, such as big data processing, AI algorithms, and interactive user experiences.

Developing Java applications for such architectures poses challenges due to limited access to hardware-level resources. Java, being a high-level language, abstracts hardware interactions and features automatic memory management, complicating direct utilization of APIs required for GPU and NPU access. To address these challenges, tools based on the Java Native Interface (JNI) and specialized libraries are employed to integrate hardware-specific computations into Java applications.

Optimizing Java applications for hybrid chips requires the use of specialized tools. Aparapi enables Java code executions on GPUs by automatically translating certain constructs into OpenCL, ensuring efficient interaction with graphical processors. Another popular solution is OpenCL for Java, which provides full access to GPU computational capabilities.

For tasks involving neural processors, TensorFlow Java API and ONNX Runtime for Java are essential tools. They allow seamless integration of machine learning models and optimize them for hardware accelerators, boosting performance in AI related tasks.

Hybrid approaches to optimization leverage technologies such as Project Panama, which facilitates direct access to native libraries and hardware resources from Java, and GraalVM, a high performance Java virtual machine with native compilation support. GraalVM also allows the creation of optimized executables that can take advantage of hybrid processor capabilities.

Conclusion. Adapting Java applications for hybrid processor architectures is a complex yet promising field in software engineering. By utilizing tools such as Aparapi, TensorFlow Java API, ONNX Runtime, Project Panama and GraalVM, developers can effectively integrate support for high-performance hardware components into Java programs. This significantly enhances the efficiency of software solutions, ensuring the full utilization of hybrid architecture resources. Future research and development of new standards in this field will further advance the adoption of such solutions.

References.

1. Aparapi: A framework for executing Java code on GPUs**.** Офіційний вебсайт. URL: https://aparapi.com (дата звернення: 20 листопада 2024 р.)
2. TensorFlow Java API documentation. URL: https://www.tensorflow.org/api_docs/java (дата звернення: 20 листопада 2024 р.)
3. ONNX Runtime documentation. URL: https://onnxruntime.ai/docs/ (дата звернення: 20 листопада 2024 р.)
4. Project Panama Overview. URL: https://openjdk.org/projects/panama/ (дата звернення: 20 листопада 2024 р.)
5. GraalVM: High-Performance Runtime for Java. URL: https://www.graalvm.org (дата звернення: 20 листопада 2024 р.)
6. OpenCL and Java integration**.** URL: https://developer.codeplay.com/opencl-java (дата звернення: 20 листопада 2024 р.)