# Kickass Enterprise Java LLM Inference

## Enterprise Application Integration 👉 Enterprise AI Integration 😅

# airhacks.industries

airhacks.live

"It's not work if you like it"
…so I never worked.  #java

# airhacks.fm

## #296 Exploring ONNX, Embedding Models, and Retrieval Augmented Generation (RAG) with Langchain4j

[episode link]

An airhacks.fm

Dmytro pre
for running
handling, e
longer text
generation
and data so
powered ap
embedding
embedding
outputs usi
specialized

Dmytro Liubars

## #313 Revolutionizing AI with Java: From LLMs to Vector APIs

[episode link] Listen on Ap

An airhacks.fm c

Alfonso previ
projects for r
multiplication
qwen or gem
for Java Com
integrating w
optimizations
but limited w
improvement
in enterprise

## #315 The AI Revolution in Java Development and Devoxx Genie

[episode link] Listen on Apple Podcasts LISTEN ON Spotify Listen on Google Podcasts [RSS]

An airhacks.fm conversation with Stephan Janssen (@Stephan007) about:

Stephan previously appeared on "#254 How JavaPolis and Devoxx Happened", discussion on the AI rev
development of an AI-assisted photo sharing application, creation of the Devoxx Genie IntelliJ plugin f
advantages of Claude 3.5 from Anthropic, use of local AI models in development environments, integra
langchain4j and its adoption by Red Hat, development of Java-based AI tools like Lama3.java, jlama an
AI models in software development, challenges and opportunities for junior and senior developers in AI
importance of understanding cloud services and cost structures when using AI, potential future of pro
generation, discussion on maintaining and improving AI-generated code, exciting developments in Jav
and tornadovm, potential for running AI models directly on Java without external dependencies, consi
and integration, the need for promoting Java's capabilities in AI development, potential for Visual Stud

airhacks.live

# airhacks.TV

with the time machine, “100 episodes ago segment”

…any questions left?

youtube.com/
@bienadam


youtube.com/bienadam/shorts


airhacks.live

NEW https://discord.gg/airhacks

# airhacks.live

## NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Winter 2024:

- Single Table Design with Java, December, 17th, 2024
- LLM / GenAI Java Bootstrap, December, 19th, 2024

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

# …I started with DevOps in 1995

then continued with serverless computing in 2001…

# Java + Clouds = ?

| Total | | | | | |
|---|---|---|---|---|---|
| | Energy | | Time | | Mb |
| (c) C | 1.00 | (c) C | 1.00 | (c) Pascal | 1.00 |
| (c) Rust | 1.03 | (c) Rust | 1.04 | (c) Go | 1.05 |
| (c) C++ | 1.34 | (c) C++ | 1.56 | (c) C | 1.17 |
| (c) Ada | 1.70 | (c) Ada | 1.85 | (c) Fortran | 1.24 |
| (v) Java | 1.98 | (v) Java | 1.89 | (c) C++ | 1.34 |
| (c) Pascal | 2.14 | (c) Chapel | 2.14 | (c) Ada | 1.47 |
| (c) Chapel | 2.18 | (c) Go | 2.83 | (c) Rust | 1.54 |
| (v) Lisp | 2.27 | (c) Pascal | 3.02 | (v) Lisp | 1.92 |
| (c) Ocaml | 2.40 | (c) Ocaml | 3.09 | (c) Haskell | 2.45 |
| (c) Fortran | 2.52 | (v) C# | 3.14 | (i) PHP | 2.57 |
| (c) Swift | 2.79 | (v) Lisp | 3.40 | (c) Swift | 2.71 |
| (c) Haskell | 3.10 | (c) Haskell | 3.55 | (i) Python | 2.80 |
| (v) C# | 3.14 | (c) Swift | 4.20 | (c) Ocaml | 2.82 |
| (c) Go | 3.23 | (c) Fortran | 4.20 | (v) C# | 2.85 |
| (i) Dart | 3.83 | (v) F# | 6.30 | (i) Hack | 3.34 |
| (v) F# | 4.13 | (i) JavaScript | 6.52 | (v) Racket | 3.52 |
| (i) JavaScript | 4.45 | (i) Dart | 6.67 | (i) Ruby | 3.97 |
| (v) Racket | 7.91 | (v) Racket | 11.27 | (c) Chapel | 4.00 |
| (i) TypeScript | 21.50 | (i) Hack | 26.99 | (v) F# | 4.25 |
| (i) Hack | 24.02 | (i) PHP | 27.64 | (i) JavaScript | 4.59 |
| (i) PHP | 29.30 | (v) Erlang | 36.71 | (i) TypeScript | 4.69 |
| (v) Erlang | 42.23 | (i) Jruby | 43.44 | (v) Java | 6.01 |
| (i) Lua | 45.98 | (i) TypeScript | 46.20 | (i) Perl | 6.62 |
| (i) Jruby | 46.54 | (i) Ruby | 59.34 | (i) Lua | 6.72 |
| (i) Ruby | 69.91 | (i) Perl | 65.79 | (v) Erlang | 7.20 |
| (i) Python | 75.88 | (i) Python | 71.90 | (i) Dart | 8.64 |
| (i) Perl | 79.58 | (i) Lua | 82.91 | (i) Jruby | 19.84 |

GraalVM™

reduces RAM footprint

https://sites.google.com/view/energy-efficiency-languages/results?authuser=0

Total

| | Energy |
|---|---|
| (c) C | 1.00 |
| (c) Rust | 1.03 |
| (c) C++ | 1.34 |
| (c) Ada | 1.70 |
| (v) Java | 1.98 |
| (c) Pascal | 2.14 |
| (c) Chapel | 2.18 |
| (v) Lisp | 2.27 |
| (c) Ocaml | 2.4 |
| (c) Fortran | 2 |
| (c) Swift | 9 |
| (c) Haskell | 0 |
| (v) C# | |
| (c) | 3.1 |
| (i) | 3.8 |
| F# | 4.13 |
| | 4.45 |
| | 7.91 |
| | 21.50 |
| | 24.02 |
| (i) PHP | 29.30 |
| (v) Erlang | 42.23 |
| (i) Lua | 45.98 |
| (i) Jruby | 46.54 |
| (i) Ruby | 6 |
| (i) Python | .88 |
| (i) Perl | 9.5 |

| | T |
|---|---|
| (c) C | 1. |
| (c) Rust | 1.0 |
| (c) C++ | 1.56 |
| (c) Ada | 1.5 |
| Java | |
| (c) Chapel | 2. |
| (c) | 2. |
| (c) | 2 |
| (c) Ocaml | 3.09 |
| C# | 3.14 |
| | 3.40 |
| Haskell | 3.55 |
| (c) Swift | 4.20 |
| (c) Fortran | 4.20 |
| (v) F# | 6.30 |
| (i) JavaScript | 6.52 |
| (i) Dart | 6.67 |
| (v) Racket | 1.27 |
| (i) Hack | |
| (i) PHP | 27 |
| (v) Erlang | 36.71 |
| (i) Jruby | 43.44 |
| | 6.20 |
| Ruby | 9.34 |
| Perl | 65.79 |
| Python | 71.90 |
| (i) | 82.91 |

| | Mb |
|---|---|
| Pascal | 1.00 |
| | 1.05 |
| (c) | 1.17 |
| (c) an | 1.24 |
| C++ | 1.34 |
| (c) Ada | 1.47 |
| (c) Rust | 1.54 |
| (v) Lisp | 1.92 |
| (c) Haskell | 2.45 |
| (i) PHP | 2.57 |
| (c) Swift | |
| (i) Python | 2.80 |
| (c) Ocaml | 2.82 |
| (v) | 85 |
| Hack | 4 |
| Racket | |
| (i) Ruby | 3.97 |
| (c) pel | 4.00 |
| (v) | 4.25 |
| (i) JavaScript | 4.59 |
| (i) TypeScript | 4.69 |
| (v) Java | 6.01 |
| (i) Perl | 6.62 |
| (i) Lua | 6.72 |
| (v) Erlang | 7.20 |
| (i) Dart | 8.64 |
| (i) Jruby | 19.84 |

partially relevant

https://sites.g e.com/v rgy-efficiency-languages/results?authuser=0

GraalVM™

reduces RAM footprint

# serverless facts

The calculations below exclude free tier discounts.

Architecture

Arm

Number of requests

2

Unit

per second

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

100

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

Value

2

Unit

GB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

Value

512

Unit

MB

▶ Show calculations

**Total Upfront cost:** 0.00 USD
**Total Monthly cost:** 15.07 USD

Show Details ▼

Cancel

Save and view summary

Save and add servi

The calculations below exclude free tier discounts.

Architecture

Arm

Number of requests

2

Unit

per second

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

10000

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

Value

2

Unit

GB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

Value

512

Unit

MB

▶ Show calculations

**Total Upfront cost:** 0.00 USD
**Total Monthly cost:** 1,402.66 USD

Show Details

Cancel

Save and view summary

Save and add servi

The calculations below exclude free tier discounts.

Architecture

Arm

Number of requests

2

Unit

per second

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

10000

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

Value

128

Unit

MB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

Value

512

Unit

MB

▶ Show calculations

**Total Upfront cost:** 0.00 USD
**Total Monthly cost:** 88.65 USD

Show Details

Cancel

Save and view summary

Save and add servi

# Who cares about “pure Java”?

# What is “Enterprise”?

# enterprise LLMs / EAI

- Explainability
- Traceability
- Cost Effectiveness
- Testability
- Security
- Privacy
- Repeatability
- Robustness

# additional challenges

- AI gateways
- Web Application Firewalls
- throttling / rate limiting
- legacy APIs / polling
- proprietary APIs
- authentication
- authorization

# LLM / AI

# remote LLM / AI

- HTTP API
- API-key authentication / RBAC
- JSON communication
- asynchronous streaming
- synchronous “RPC”
- …just another, unreliable and slow, microservice

# unusual nature

- idempotent (mostly)
- stateless API
- stateful communication
- slow (high latency)
- low availability
- defined scalability
- robust
- IO bound

# Anthropic Claude

```
public interface Claude {

    enum Models{
        CLAUDE_35_SONNET("claude-3-5-sonnet-20240620"),
        CLAUDE_30_OPUS("claude-3-opus-20240229");
        private String modelName;
    }

    HttpClient client = HttpClient.newHttpClient();
    URI uri = URI.create("https://api.anthropic.com/v1/messages");
    String ANTHROPIC_VERSION = "2023-06-01";
    String API_KEY = "duke";
    Models currentModel = Models.CLAUDE_35_SONNET;


    public static JSONObject invoke(String system, String user, float temperature){
        var enclosedPrompt = messagePrompt(user);
        var payloadJSON = Claude.claudeMessage(enclosedPrompt,temperature,system);
        payloadJSON.put("model", currentModel.modelName());
        var payload = payloadJSON.toString();
        var answer = invoke(payload);
        return new JSONObject(answer);
    }

    static JSONArray messagePrompt(String user) {
        return new JSONArray()
                .put(message("user", user));

    }

    static JSONObject message(String role, String content) {
        return new JSONObject()
                .put("role", role)
                .put("content", content);
    }


    static JSONObject claudeMessage(JSONArray messages, float temperature, String system) {
        return new JSONObject()
                .put("max_tokens", MAX_TOKENS)
                .put("messages", messages)
                .put("temperature", temperature)
                .put("system", system);

    }

    static String invoke(String message) {
        var request = HttpRequest.newBuilder(uri)
                .POST(BodyPublishers.ofString(message))
                .header("x-api-key", API_KEY)
                .header("content-type", "application/json")
                .header("anthropic-version", ANTHROPIC_VERSION)
                .build();
        try {
            var response =  client.send(request, BodyHandlers.ofString());
            var body = response.body();
            if(response.statusCode() == 529){
                Log.error("claude is overloaded, please try again later " + body);
            }
            return body;
        } catch (IOException | InterruptedException e) {        }
    }
}
```

# Claude via AWS Bedrock

```
public interface Bedrock {
        String BEDROCK_VERSION = "bedrock-2023-05-31";
        enum Model {
                CLAUDE_V21("anthropic.claude-v2:1", Region.EU_CENTRAL_1),
                CLAUDE_V3_SONNET("anthropic.claude-3-sonnet-20240229-v1:0", Region.US_EAST_1);
        }

        Model model = Model.CLAUDE_V3_SONNET;
        AwsCredentials credentials = AwsBasicCredentials.create("chief","duke");
        static String getPrompt(String system, String user) {
                if (system == null || system.isEmpty()) {
                        return "\n\nHuman: %s\n\nAssistant:".formatted(user);
                } else {
                        return """
                                        %s
                                        \n\nHuman: %s\n\nAssistant:
                                        """.formatted(system, user);
                }
        }
        public static JSONObject invokeClaude(String user, float temperature) {
                return invokeClaude(null, user, temperature);
        }

        public static JSONObject invokeClaude(String system, String user, float temperature) {
                var client = bedrockRuntimeClient();
                var enclosedPrompt = Claude.messagePrompt(user);
                var payloadJSON = Claude.claudeMessage(enclosedPrompt,temperature,system);
                payloadJSON.put("anthropic_version", BEDROCK_VERSION);
                var payload = payloadJSON.toString();
                var request = InvokeModelRequest.builder()
                                .body(SdkBytes.fromUtf8String(payload))
                                .modelId(model.modelId())
                                .overrideConfiguration(AwsRequestOverrideConfiguration.builder()
                                                .apiCallTimeout(Duration.ofMinutes(5))
                                                .build())
                                .contentType("application/json")
                                .accept("application/json")
                                .build();
                var response = client.invokeModel(request);
                var message = response.body().asUtf8String();
                return new JSONObject(message);
        }

        private static BedrockRuntimeClient bedrockRuntimeClient() {
                var client = BedrockRuntimeClient
                                .builder()
                                .region(model.region())
                                .httpClientBuilder(UrlConnectionHttpClient
                                                .builder()
                                                .socketTimeout(Duration.ofMinutes(5))
                                                .connectionTimeout(Duration.ofMinutes(5)))
                                .overrideConfiguration(ClientOverrideConfiguration
                                                .builder()
                                                .apiCallTimeout(Duration.ofMinutes(5))
                                                .build())
                                .credentialsProvider(StaticCredentialsProvider.create(credentials))
                                .build();
                return client;
        }
}
```

# quarkus / langchain4j

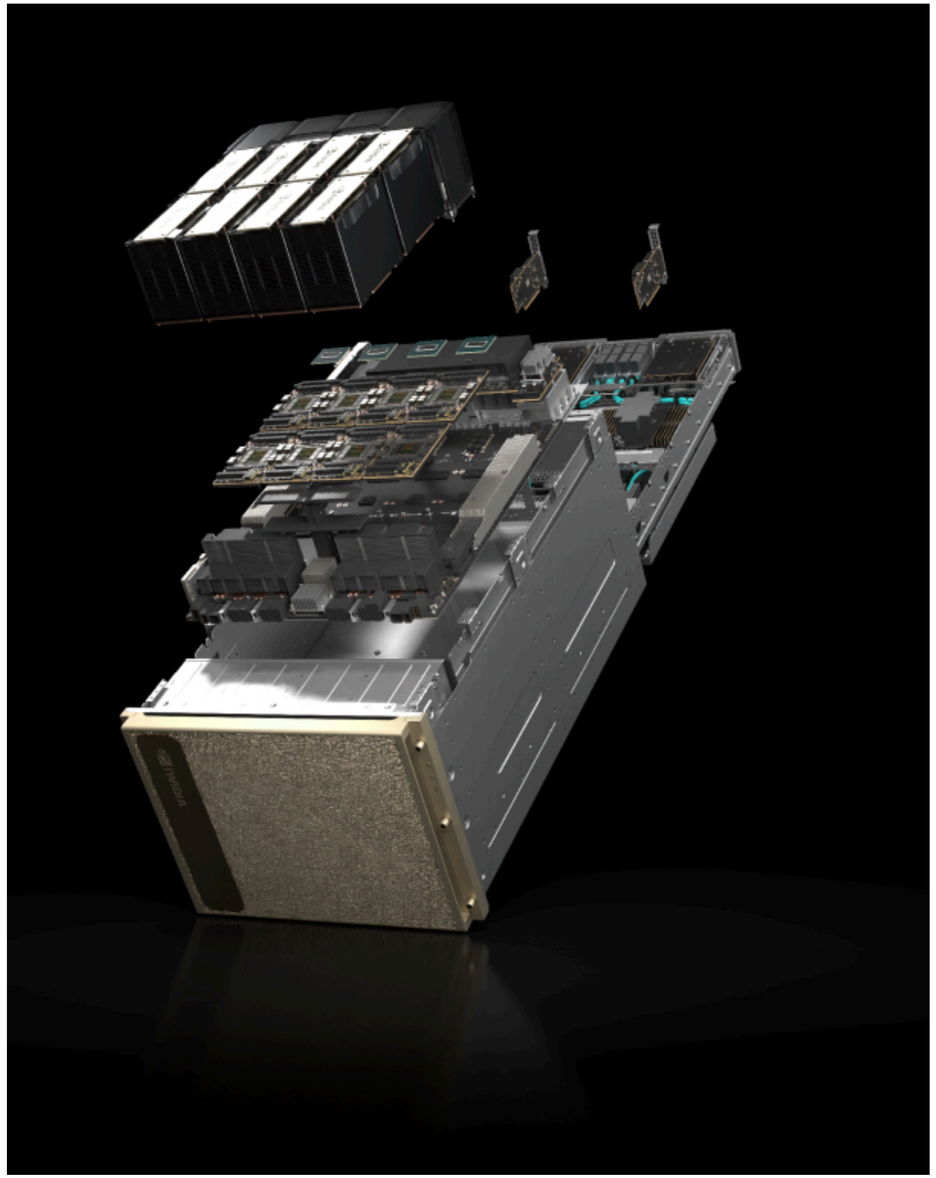
```java
@RegisterAiService
public interface TriageService {

    @SystemMessage("""
            You are working for a bank. You are an AI processing reviews about financial products
            You will always answer with a JSON document, and only this JSON document.
            """)
    @UserMessage("""
            Your task is to process the review delimited by ---.
            Apply a sentiment analysis to the passed review to determine if it is positive or neg
            The review can be in any language. So, you will need to identify the language.

            For example:
            - "I love your bank, you are the best!", this is a 'POSITIVE' review
            - "J'adore votre banque", this is a 'POSITIVE' review
            - "I hate your bank, you are the worst!", this is a 'NEGATIVE' review

             Answer with a JSON document containing:
            - the 'evaluation' key set to 'POSITIVE' if the review is positive, 'NEGATIVE' otherw
            - the 'message' key set to a message thanking the customer in the case of a positive

            ---
            {review}
            ---
            """)
    TriagedReview triage(String review);
```

# local / offline LLM / AI

- Docker e.g. Ollama
- ONNX / GGUF direct integration
- JNI / Vector API / Panama
- HTTP API to localhost
- memory and CPU and GPU hungry
- behaves like another legacy microservice (e.g. PL1 / COBOL integration)
- unstable
- high latency

# EAI (Enterprise Application Integration 😀) 2.0?


VS.



https://www.nvidia.com/en-us/data-center/dgx-h200/

# offline challenges

- native access
- CI/CD, IaC, provisioning
- concurrency
- garbage collection, memory management
- provisioning, Infrastructure as Code
- hardware costs

# Hexagonal Architectures

DEEP NETTS

bedrock

claude

langchain4j

java application

Meta

llama

jlama

llama3.java


ChatGPT

# Patterns (Ideas) Revival

- Adapter / Decorator (caching)
- DAOs (RAG)
- Inversion of Control (tools)
- Fluid Logic
- API / SPI
- UnexpectedFormatException
- HallucinationException

# throwback

- ConversationalScope
- SessionScope
- BeanValidation
- Parameterized Unit Tests (temp, top-k)
- Integration Tests
- Java Connector Architecture / JCA (langchain4j)
- Bean Validation

# The Fat Enterprise “AI” JAR
# 👉FAIAR🚀

# LLM👉just another unstable service

```
using claude model: claude-3-5-sonnet-20240620
claude is overloaded, please try again later {"type":"error","error":{"type":"overloaded_error","message":"Overloaded"}}
content key not found in: {"type":"error","error":{"type":"overloaded_error","message":"Overloaded"}}
```

# Fault Tolerance

# The Remedy

@Timeout

@Retry

@Fallback

@CircuitBreaker

@Bulkhead

@Asynchronous

…and interceptors

# aspects

result caching

call tracing

“load balancing”

“cost balancing”

# What about langchain4j?

# airhacks.live

## NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Winter 2024:

- Single Table Design with Java, December, 17th, 2024
- LLM / GenAI Java Bootstrap, December, 19th, 2024

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

# Thank YOU!

airhacks.industries

airhacks.live