UVSQ ISTY
université PARIS-SACLAY
Institut des Sciences et Techniques des Yvelines
CAMPUS DE MANTES EN YVELINES
CAMPUS DE SAINT-QUENTIN-EN-YVELINES

10-12 Av de l'Europe, 78140
Vélizy-Villacoublay, France

aneo

122 Av Général Leclerc,
92100 Boulogne-Billancourt, France

Referent teacher:
Dhekra ABOUDA-FOUGEROUSSE

Tutor:
Florian LEMAITRE

INTERNSHIP REPORT

Engineering school internship: May 2024 – August 2024

DEVELOPMENT OF MONITORING TOOL TO ANALYSE RESULT CREATION IN THE ARMONIK ORCHESTRATOR

Nylls Gersan BOUTOTO - Student computer engineer, *"IATIC 4"*

Acknowledgements

I would like to express my deep gratitude to the ArmoniK team for their guidance and support throughout the first eight weeks of my internship.

Special thanks go to my internship tutor, Florian Lemaitre, for his constant support and for giving me the opportunity to work on a subject allowing me to express my passion for algorithms and apply the graph theory concepts I had learned in class.

And a big thank you to all the readers of this report, especially Nicolas Gruel for following and supporting me throughout the writing of this report.

Content

Table of figures

Glossary

Here is an alphabetically sorted list to help you understand every term you might encounter in this report.

API: Application Programming Interface

HTC (High Throughput Computing): A computing paradigm focused on processing a large number of tasks that can be executed independently.

HPC (High Performance Computing): A computing paradigm that involves the use of supercomputers and parallel processing techniques for solving complex computational problems.

Kubernetes: An open-source platform for automating the deployment, scaling, and management of containerized applications.

RSE: ‘’Responsabilité Sociétale des Entreprises.’’

Subtasking: In the context of High-Throughput Computing (HTC) with ArmoniK refers to the process of dividing a large computational task into smaller, more manageable subtasks that can be executed in parallel. This approach allows for more efficient use of computing resources, reducing overall execution time, and improving performance.

URL: Uniform Resource Locator.

Introduction

During my internship at ANEO, I had the chance to work on improving ArmoniK, an open-source software capable of managing intensive High Throughput Computing (HTC) workloads, as well as HPC (High Performance Computing) workflows. ArmoniK is designed to be resilient, elastic, and scalable. It uses Kubernetes to be deployed on different environments, whether in the cloud or on-premises. ArmoniK has been in production for two years and allows its clients to distribute millions of computing tasks across thousands of nodes/pods, adjusting resources based on the workload.

Armonik's scheduling model is based on a graph of tasks and data. Users define the links between tasks and data when submitting the graph, ArmoniK can also let the computation tasks change the graph dynamically. This flexibility is powerful but makes it sometimes difficult to track the progress of a specific result.

The main goal of this internship was to develop a tool to track the progress of a result production in ArmoniK.

To achieve this goal, my tasks are to propose a low-intrusive approach to continuously rebuild and update the graph that the result depends on and identify tasks or data that block the progress of an execution. Depending on the progress of the internship, the work could potentially expand to implementing a new orchestration algorithm based on lazy evaluation of the task graph, as well as creating a web interface for visualizing progress and diagnosing issues.

I. ANEO

1. ANEO: Overview & Some Figures

ANEO is a consulting company that helps businesses with digital and organizational changes. Founded in 2002 by Pierre Sinodinos, the company has gone through several restructurings and strategy changes, making ANEO a unique company. Today, it has between 100 and 199 employees, and its revenue for 2023 is 23,612,700 euros.

From its creation and until 2009, the company was seen as an IT services company, focusing on new technologies and computing. After that, ANEO started to change its offers, facing some organizational challenges. Eventually, ANEO became an "hybrid" company. This means that, in addition to digital services, it also offers advice in IT, project management, and business management.

One of the unique points about ANEO is how it manages its employees. The company uses a “flat hierarchy”, which means the chain of command is kept to a minimum. This makes administrative processes simpler by reducing the number of intermediaries. Most of the time, interactions happen directly between the people involved. This helps all employees feel valued and easily accepted in the company, greatly improving relationships and communication.

2. Circles and colors

ANEO employees are divided into three main branches:

Figure 1: Aneo's circles

- ✓ **Strategy**: This branch handles strategy for organizations, information systems, and user experience.
- ✓ **Management**: Specializes in value-driven management, organizational agility, and managerial transformation.
- ✓ **Technology**: Works on the company’s HPC Cloud, software process industrialization, and data and artificial intelligence topics.

These three branches are the core of ANEO. Inside the company, we often talk about "yellow" and "red" members. The "red" members work on technology topics, and the "yellow" members do everything else (management, design…). Now, ANEO wants to merge these two groups into one. This would allow all employees to share their skills and improve their relationships. This new group would be "orange," the color you get by mixing yellow and red, which is also in ANEO’s logo.

3. Events

ANEO also organizes various events with different goals. These include welcoming new employees, sharing company news, and providing training sessions between coworkers.

a. ANEO Lunch

Figure 2: ANEO Lunch

The first of these events happens on Fridays, every two weeks. All ANEO employees are invited to join, at least via video conference. The ANEO Lunch is divided in two parts:

At first, some members of the executive team talk about the company's updates. This starts with introducing new members, whether they are interns, apprentices, or employees. They introduce themselves to everyone present.

Next, the company's RSE team explains the different initiatives ANEO has started. This includes “La fresque du climat” and “La fresque du numérique” mural, which are important topics for raising awareness about ecology, often discussed during these meetings.

The next topic is about ANEO's new projects, new contracts, and the employees who will work on them. A brief description of each mission is given so everyone understands what it is about. This is done to encourage the sharing of experiences among employees.

To finish this first part of the ritual, the upcoming internal events are presented, whether they are related to wine tasting, video games or others!

The second part of the ANEO Lunch is just a mealtime, where employees on-site can talk about any topic and interact together.

b. ANEO - Seminars, Company outings

Moreover, I also had the opportunity to participate in various seminars, both for new arrivals and expertise, linked to the circle I belong to. Among them, the seminar on "La fresque du climat" particularly impressed me. I find this topic crucial because it is at the heart of current social issues and is very important today.

Figure 3: Clim'A6

Additionally, I had the opportunity to attend two company external activities: first, to assist to the forum Teratec 24, dedicated to the French HPC communities. I went there with the ArmoniK team which I belong to,

Figure 4: Output Teratec

and secondly, the celebration of ANEO’s anniversary.

Figure 5: ANEO’s aniversary

4. Living and Working Environment

ANEO is not just about the events it organizes for its employees. Other elements help to mix social circles, like the office layout and a system very similar to what you can find in most higher education institutions.

a. ANEO’s Offices - Dynamic Spaces

ANEO's offices in Boulogne-Billancourt are organized in a unique way, expanding upon the principles of open space. They operate on a flex-office basis, meaning no seats are reserved for anyone. Of course, the executive team and employees working with sensitive data have assigned seats exceptionally. Apart from these, anyone can choose any seat, change their habits for a week or a month, and most importantly, sit next to people working in different fields. This blend is a strong reflection of ANEO's identity.

Colors are also a part of ANEO's identity, and this is evident in the meeting rooms throughout the premises. Each meeting room is associated with a specific color.

Figure 6: Flex Office

b. Clubs

To conclude on integrating employees into the company, I would like to mention the clubs offered here. These clubs provide activities related to various interests. Thus, employees from different circles are once again mixed together, this time around common interests. Among these clubs, there is of course one dedicated to video games, as well as those focused on oenology, climbing, and even zythology.

I was pleasantly surprised by the presence of these clubs, which I did not expect to find in a professional environment. While I understand that not every company has such offerings, I am delighted by this discovery!

II. Internship Framework

As mentioned earlier in the introduction, I had the opportunity to work on the ArmoniK project during my internship, specifically on the execution graph (task and data graph).

1. ArmoniK team

First of all, the ArmoniK project is managed by the ArmoniK team, which I joined upon my arrival. The ArmoniK team into five small teams, each having a different role. There are the **Sales** team, **Product Development**, **Integration, Support,** and the **Internal Monitoring** team.

Now let’s delve a bit deeper into each of them:

Sales mainly involves finding and contacting new clients, especially to help finance the development of ArmoniK. **Product development** focuses on new features planned by the development team. The **Integration** team helps clients integrate ArmoniK into their workflow: configuration, production, testing... This team can also modify ArmoniK's source code to fit the clients' needs. They also pass client requests to the development team. The **Support** team helps clients with problems or specific needs. They are the first ones contacted in case of issues. Finally, the **Internal Monitoring** team ensures the project is progressing in the right direction and at an acceptable pace, making sure goals are met on time.

The team follows an agile methodology designed especially for this project and inspired by kanban methods. The most known agile method, scrum, with its 2 to 4 weeks sprints, was not chosen. The team felt that fixed sprints were a burden for developers and slowed down development. Instead, the most experienced members created their own agile method. Instead of a Product Owner, Tech Leads talk directly to clients. Sprints were replaced by deliveries when the code is ready, about once a month, if the code is ready. This allows developers to reduce indirect communication between clients and other managers. The development team can accept or reject client requests faster, avoiding unnecessary discussions. Removing sprints lets developers work without pressure, ensuring code quality. This method is based on trust: if a developer works too little, it can affect the whole team.

2. Familiarization with ArmoniK software

After getting acquainted with the organization and workflow of the ArmoniK team, I will now turn my attention to the ArmoniK software itself. Understanding the software was crucial for effectively contributing to the project. Here's how I began to familiarize myself with its various aspects.

a. ArmoniK in brief

As a reminder, my first task was to continuously rebuild and update the ArmoniK's tasks and data graph on which the expected result depends. Then, I had to identify the tasks or data blocking the progress of the execution. ArmoniK, a high throughput compute grid project using Kubernetes, provides a reference architecture for building and adapting modern high throughput compute solutions both on-premise and in the Cloud. It allows users to submit high volumes of short and long-running tasks while dynamically scaling environments. However, after discussing with my internship supervisor, we agreed that I should focus on the static aspect of the graph rather than its dynamic aspect. In other words, I needed to be able to rebuild the graph after each execution and analyze it afterward. Consequently, continuous updating was set aside for now, and I focused on post-execution analysis.

b. ArmoniK Deployment

One of the initial expectations set by my internship supervisor after our first meeting and his presentation of the ArmoniK application was for me to conduct various ArmoniK tests, such as HTC mock, Bench, and HelloWorld, to familiarize myself with the software. Additionally, it was crucial for me to have a solid understanding of terms like sessions, tasks, and results, as they play a pivotal role in the project's progress. We also agreed to meet once a week, on Wednesdays, to monitor my progress step by step. To perform these tests, the first step was to deploy the ArmoniK application. This process involved meticulously installing the ArmoniK repository and its dependencies as outlined in the documentation. Deploying ArmoniK was a completely new experience for me, which made it both challenging and exciting. So, what steps are involved in deploying ArmoniK?

Firstly, ArmoniK can be deployed locally on Linux, on an on-premise cluster, or on a cluster provided by cloud services such as Google Cloud Platform (GCP) or Amazon Web Services (AWS), which are supported by ArmoniK. It can also be deployed on Windows using a Linux virtual machine, like WSL2, which is what I did.

To do this, after cloning the ArmoniK remote repository onto my virtual machine WSL2, I had to proceed to install the prerequisites in order to enable local deployment.

The installation of ArmoniK prerequisites is done notably with running of the following script: ./infrastructure/utils/scripts/installation/prerequisites-installer.sh.

It installs the following libraries:

- **Make**: ArmoniK uses Make to execute diverse Makefiles instructions.
- **JQ**: ArmoniK uses JQ to parse JSON.
- **Python3**: ArmoniK uses Python3 to run scripts.
- **HCL2**: ArmoniK uses HCL2 to parse Terraform files.
- **Jsonpath-ng**: ArmoniK uses jsonpath-ng to parse JSON.
- **Terraform**: ArmoniK uses Terraform to deploy infrastructure.
- **Docker**: ArmoniK uses Docker to run containers.
- **Helm**: ArmoniK uses Helm to deploy applications.
- **Kubernetes**: ArmoniK uses Kubernetes to orchestrate containers.
- **K3s**: ArmoniK uses K3s, a lightweight Kubernetes distribution, for development environments.
- **Kubectl**: ArmoniK uses kubectl to manage Kubernetes.

Next, to deploy the ArmoniK application, we navigate to the infrastructure/quick-deploy/localhost directory of the repository, then enter the command: “make or make deploy”. This command will run every needed step to deploy the application.

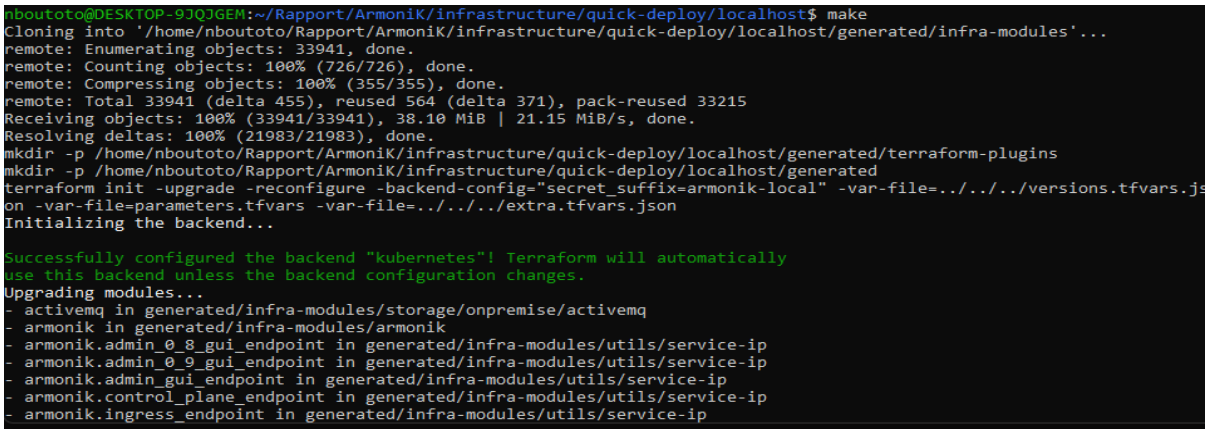

Figure 7: ArmoniK’s deployement

c. ArmoniK tests: HTC Mock, Bench, and HelloWorld

After deployment, the next step is to run the various tests on the ArmoniK application to make sure everything was working properly. First of all, I started with launching HTC mock. HTC Mock is a test tool for ArmoniK used to generate tasks that will be run by the ArmoniK framework. It is also employed in Continuous Integration (CI) to automatically test ArmoniK, but it can also be utilized during development to generate realistic tasks. The particularity of HTC mock is that it allows for subtasking.

Next, I launched Bench. It is somewhat similar to HTC mock but does not support subtasking. Bench is specifically used for performance regression testing, which means checking if there has been a loss or gain in performance from one version to another.

And, finally, I launched a HelloWorld. First of all, I needed to clone the ArmoniK.Samples repository in addition to the ArmoniK repository. HelloWorld is really a more concrete example, compared to

HTC mock or Bench, because to run HelloWorld, we have to write our own partition, what I did following the documentation provided.

d. ArmoniK in Practice

ArmoniK's architecture follows a client-worker model, which is similar to the client-server model used in network or web applications.

- Client: User-developed software that submit tasks and data graphs to ArmoniK and retrieve results.
- Worker: User-developed software capable of performing one or several tasks depending on its implementation:
    - Can simply take input data and perform calculations on it to return result.
    - Can submit new tasks that will self-performed, or by different workers, other instances of itself.

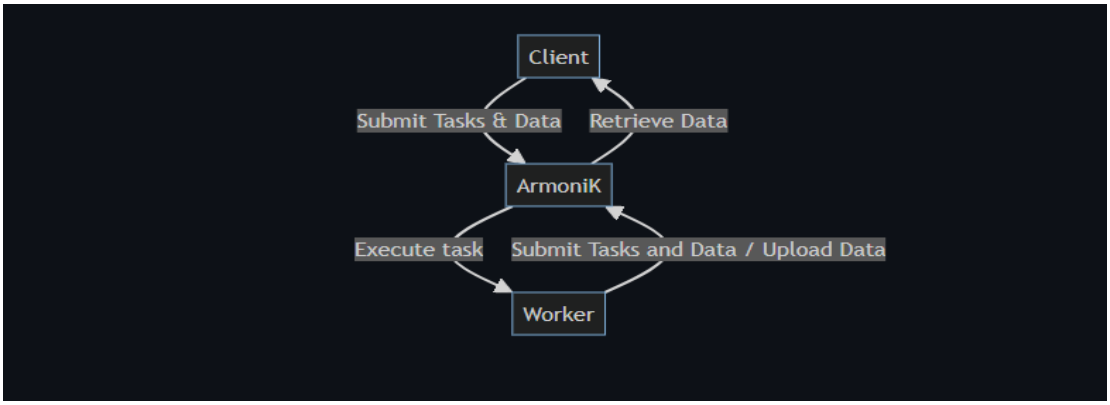

Figure 8: Client-Worker model's ArmoniK

In practice, when we submit the tasks and data graph, we create a session. A **session** logical group **tasks** and **results**, concretely a session will consist of a set of tasks, each responsible for a computation and a key component of ArmoniK. Tasks return results, which can be used as input for other tasks. Sessions can be executed in different environments, called **partitions**.

Partitions are sets of pods with their own configuration, their own task queue, and their own scaling rules. Partitioning is useful in the following use-cases:

- Workers need different images docker
- Workers need different node configuration (eg: number of cores, gpu)
- Need different limits on the number of pods depending on the application.

3. Execution graph reconstruction

This task was part of a context where it was necessary to monitor the progress of result production after tasks execution. To do this, it was imperative that I familiarize myself with the ArmoniK Python API as well as the rustworkx library.

a. Preparation and Familiarization

Before starting the practical work, I dedicated time to understanding the features of the ArmoniK API. This API is essential to interact with the ArmoniK system, particularly for retrieving sessions and associated tasks. At the same time, I explored the rustworkx library, a powerful tool for manipulating and visualizing graphs in Python. This dual familiarization was crucial for the continuation of my project.

b. Initialization of Git Repository

To structure and track the progress of my work, I initialized a Git repository at the beginning of the project. This repository served as a central reference to store and manage all the code I developed. By using Git, I could track changes, manage versions, and collaborate effectively with my supervisor in the objective is to provide a Python library to track the progress of result production within ArmoniK.

c. Design of the Module

Within the Git repository, I notably worked on a module allowing the reconstruction of the execution graph after tasks have been executed during a given session in ArmoniK, and subsequently analyzing it, I have developed three key functionalities so far:

retrieve_sessions_for_endpoint: This function retrieves a list of sessions from a specified ArmoniK endpoint. It establishes a communication channel using the endpoint URL, initializes an ArmoniKSessions object, and retrieves all sessions associated with the endpoint. It returns a list of Session objects representing these sessions.

retrieve_tasks_for_session: This function retrieves a list of tasks associated with a specific session from an ArmoniK endpoint. It takes the endpoint URL and the session ID as arguments, establishes a channel using insecure_channel, and initializes an ArmoniKTask object. It filters tasks based on the provided session ID using TaskFieldFilter, retrieves these tasks, and returns a list of Task objects representing them.

reconstruct_execution_graph: This function reconstructs the execution graph within a given session based on a provided list of tasks (Task objects). It initializes a PyDiGraph object to represent the graph and maps out dependencies between tasks and their input/output data. Nodes in the graph represent either tasks or input/output data, while edges signify dependencies between them.

d. Unit Testing with Pytest: Fundamentals and Application

What is a Unit Test?

It is not unusual, but I started by implementing the functionalities without writing unit tests first. This approach diverged from the recommended practice of the Tests Driven Development approach. In practice, unit tests serve as automated procedures to validate that code units, such as functions, methods, or classes, yield expected outputs for diverse inputs. Their primary objective is to verify the isolated functionality of each piece of code, ensuring it operates correctly independently from the rest of the program.

Why Perform Unit Testing?

Unit tests offer several crucial advantages. They enable early error detection, helping to catch bugs as soon as they appear and facilitating their correction before they affect other parts of the software. Additionally, unit tests ease maintenance by isolating errors to specific units, making refactoring and

code modifications safer and less risky. Furthermore, unit tests serve as living documentation by describing the expected behavior of each unit of code.

Using Pytest

During my internship, I used Pytest, a popular unit testing framework in Python that simplifies writing and executing tests with its concise syntax and advanced features. I learned to write unit tests using Pytest after exploring the basics of unit testing. For instance, I implemented tests to validate a function's behavior with valid inputs, check the function's response when provided with an empty endpoint, and verify the function's behavior with an empty session ID. Pytest allows for defining clear assertions and catching expected exceptions, ensuring that the function behaves as expected in different scenarios. Additionally, Pytest facilitates running all defined tests in a project with a simple command and provides detailed reports on test results, helping to quickly identify potential failures.

In summary, moving forward in this internship, I will first need to write tests before implementing the functionalities.

Conclusion

In English

These first eight weeks of work experience at ANEO have been extremely rewarding and have laid a solid foundation for the rest of the project. The main achievements at this stage are varied and significant. Firstly, I was able to familiarize myself in depth with the architecture and operation of ArmoniK. Secondly, I initially developed a Python module designed to reconstruct the task execution graph. In addition, I implemented key functionalities, such as session and task retrieval. Finally, I set up unit tests using Pytest.

This work is an excellent starting point for the remaining eight weeks of the placement. The next steps are promising and ambitious. I will be finalizing and optimizing the execution graph reconstruction module. I will also be developing graph analysis functions to identify bottlenecks. There are also plans to potentially start work on a visualization interface.

Aside from the technical aspect, this first half of my placement has also enabled me to integrate into ANEO's unique corporate culture, with its flexible organization and numerous events that encourage exchanges. It's already been a very formative experience, both professionally and personally. The next eight weeks look set to be just as rewarding and should enable me to consolidate what I've learnt while learning new things.

In french

Ces huit premières semaines de stage chez ANEO ont été extrêmement enrichissantes et ont permis de poser des bases solides pour la suite du projet. Les principales réalisations à ce stade sont variées et significatives. Tout d'abord, j'ai pu me familiariser en profondeur avec l'architecture et le fonctionnement d'ArmoniK. Ensuite, j'ai développé initialement un module Python destiné à reconstruire le graphe d'exécution des tâches. De plus, j'ai implémenté des fonctionnalités clés, telles que la récupération des sessions et des tâches. Enfin, j'ai mis en place des tests unitaires avec Pytest.

Ces travaux constituent un excellent point de départ pour les huit semaines restantes du stage. Les prochaines étapes envisagées sont prometteuses et ambitieuses. Il s'agira de finaliser et optimiser le module de reconstruction du graphe d'exécution. Par ailleurs, je développerai des fonctionnalités d'analyse du graphe afin d'identifier les goulots d'étranglement. Il est également envisagé de potentiellement commencer le travail sur une interface de visualisation.

Au-delà de l'aspect technique, cette première moitié de stage a également permis de m'intégrer dans la culture d'entreprise unique d'ANEO, avec son organisation flexible et ses nombreux événements favorisant les échanges. Cette expérience est déjà très formatrice, tant sur le plan professionnel que personnel. Les huit semaines à venir s'annoncent tout aussi enrichissantes et devraient permettre de consolider les acquis tout en relevant de nouveaux défis pour le projet ArmoniK.

Webography/Bibliography

[1] ArmoniK – Multiverse 2024, Product Presentation “The alternative to your Grid Orchestrator” by ArmoniK Team

[2] The client-worker pattern: Formations/armonik-rd.md at main · aneoconsulting/Formations (github.com)

[3] Digital information about ANEO: https://www.societe.com/

[4] ArmoniK official website: https://www.armonik.fr/

[5] Information about Kubernetes: https://kubernetes.io/

[6] Information about pytest : pytest: helps you write better programs - pytest documentation

[7] Information about rustworkx : rustworkx 0.15.1