AI-RAN Platform and Infrastructure Orchestrator

AI-RAN Alliance

WG2 AI-and-RAN

ai-ran.org

AI-RAN™ ALLIANCE

Contributors (in alphabetical order, first name):

Emmanuel Marilly (Nokia)

Hyde Sugiyama (Red Hat)

Jun Song (Samsung)

Kuntal Chowdhury (Nvidia)

Martin Julien (Ericsson)

Michele Polese (Northeastern University)

Niloofar Mohamadi (zTouch Networks)

Shinta Sugimoto (SoftBank)

Shuvo Chowdhury (Nvidia)

Editor

Shinta Sugimoto (SoftBank)

Contents

1. Introduction 5

1.1. Background and context 5

2. Assumptions and Prerequisites 5

2.1. Alignment with the architecture 5

2.2. AI-and-RAN 5

2.3. RAN Workloads 8

2.4. AI Workloads 8

2.5. AI Accelerators 9

3. Related Work 9

3.1. ETSI MANO NFVO 9

3.2. O-RAN SMO 9

3.3. Kubernetes and Open Cluster Management 10

3.4. Nephio 10

3.5. ONAP 10

4. Problem Statements 11

5. Use Cases 11

5.1. Use Case 1: Intelligent and Constraint-Aware Workload Placement 11

5.2. Use Case 2: Dynamic Resource Allocation and Reconfiguration 11

5.3. Use Case 3: SLA-driven Resource Preemption and Prioritization 12

5.4. Use Case 4: External AI Workload Execution 12

5.5. Use Case 5: Predictive and Proactive Orchestration 12

6. Requirements 13

6.1. Functional requirements 13

6.2. Non-functional requirements 15

7. System Architecture 16

7.1. Network Overview 16

7.2. Functional Components 16

7.3. Orchestrator Processes 18

8. AI-RAN P&I Orchestrator Interfaces 20

8.1. North Bound Interface 20

8.2. East/West Bound Interface 20

8.3. South Bound Interface 20

8.4. Role Separation from Domain Specific Orchestrator 21

9. Further Study 22
10. Glossary of Terms 23
11. References 24
12. Document History 25

1Introduction

1.1. Background and context

The evolution of AI technology has been remarkable, with generative AI-based applications and services rapidly emerging. The development of AI, particularly large language models, requires immense computational power, primarily on GPUs. The demand for computing resources continues to grow for both AI training and inference.

Cloud computing significantly reduces CAPEX by enabling software to run on COTS hardware. The transition from traditional network functions to virtualized network functions (VNFs) and cloud-native network functions (CNFs) has been taking place globally. Radio Access Networks (RANs) are no exception. The O-RAN ALLIANCE defines the architecture for next-generation RAN under the assumption that many RAN components, such as CUs and DUs, are virtualized or containerized. The adoption of virtualized and cloud-native network functions introduces a key shift: compute resources are no longer dedicated solely to running RAN workloads.

RAN exhibits dynamic characteristics, as its usage depends on human activity. It operates at full capacity during peak user activity and remains idle during off-peak hours. To maximize the utilization of computing resources, accurate prediction of RAN compute requirements is highly desirable.

2. Assumptions and Prerequisites

2.1. Alignment with the architecture

This document defines the AI-RAN Platform and Infrastructure (P&I) Orchestrator as outlined in the AI-RAN architecture document [1]. The AI-RAN P&I Orchestrator defined in this document also corresponds to the E2E Orchestrator at OL2 in the AI-and-RAN architecture document. Note also that the AI-RAN P&I Orchestrator defined in this document primarily focuses on orchestrating and managing resources at the container platform layer (i.e., Kubernetes). Future iterations of this document will focus on infrastructure orchestration.

2.2. AI-and-RAN

The concept of “AI-and-RAN” means using a common IT infrastructure to accommodate both AI and RAN workloads. Therefore, there is a fundamental requirement for the infrastructure to run AI and RAN workloads concurrently. It is important to note that AI and RAN workloads may have different SLAs. Concurrently, there are opportunities to reposition IT infrastructure as a revenue-generating asset by executing AI workloads. The key to achieving the goal of “AI-and-RAN” is to deploy and manage compute, networking, and storage resources so that both AI and RAN

workloads can meet their respective performance and resource requirements while running simultaneously on the same infrastructure. The AI-RAN Platform and Infrastructure (P&I) Orchestrator can dynamically configure the infrastructure as software, without hardware updates, to change the type of workload supported by the infrastructure.

The platform and infrastructure resource management can be performed at varying levels of granularity, resulting in different management scenarios:

- **Scenario A** – AI-RAN infrastructure servers can run both AI and RAN workloads concurrently by sharing AI accelerators, CPUs, and other resources (e.g., RAM/networking).
- **Scenario B (Per server)** – In this scenario, the types of workloads to be run are isolated per server. For example, suppose a cluster consists of 8 worker nodes (worker nodes 01, 02,..., 07). The AI-RAN P&I Orchestrator configures some of them (e.g., worker nodes 01, 02, and 03) for RAN, whereas the rest (worker nodes 04, 05, 06, and 07) are configured for AI. The AI-RAN P&I Orchestrator can change this configuration over time.
- **Scenario C (Per cluster)** – In this scenario, the types of workloads to be run are isolated per cluster. For example, suppose that a data center consists of multiple clusters (cluster 1, cluster 2, and cluster 3). The AI-RAN P&I Orchestrator configures cluster 1 and 2 to run RAN workloads and cluster 3 to run AI workloads. The AI-RAN P&I Orchestrator can change this configuration over time.

Figures 1 through 3 illustrate examples of resource management in their respective scenarios.

- Green-colored objects represent clusters or nodes configured for AI, as well as AI tasks, while blue-colored objects represent clusters or nodes configured for RAN, as well as RAN tasks.
- The arrow in Figure 1 indicates a CPU affinity setting, meaning that a given task is assigned to a specific CPU core (i.e., CPU pinning).
- In Scenario A, partitioning of the CPU and AI accelerator is crucial. For CPUs, partitioning can be achieved through techniques such as CPU isolation and CPU affinity configuration (i.e., CPU pinning). An example of technology to partition GPU resources is NVIDIA's Multi-Instance GPU (MIG).

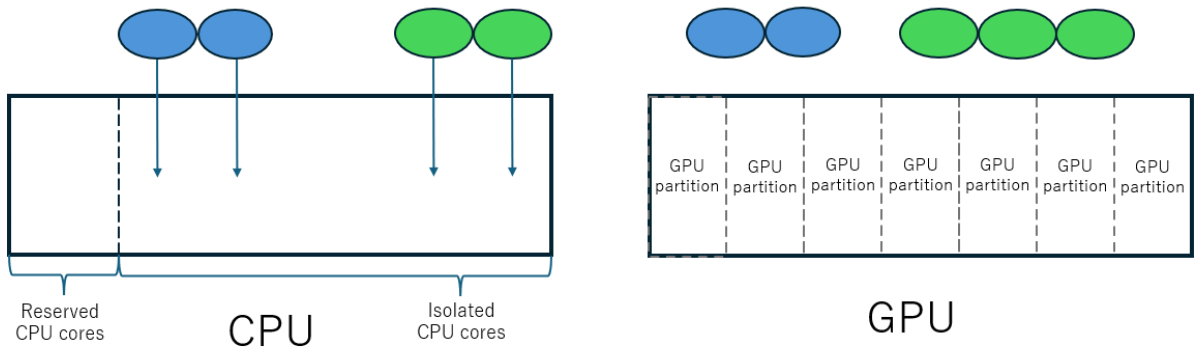

Figure 1: An example of Scenario A

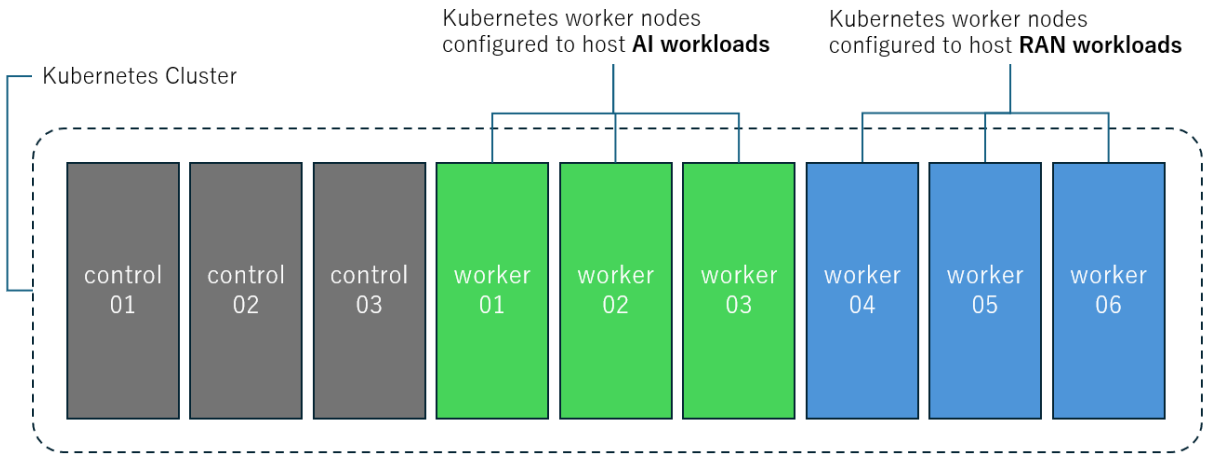

Figure 2: An example of Scenario B

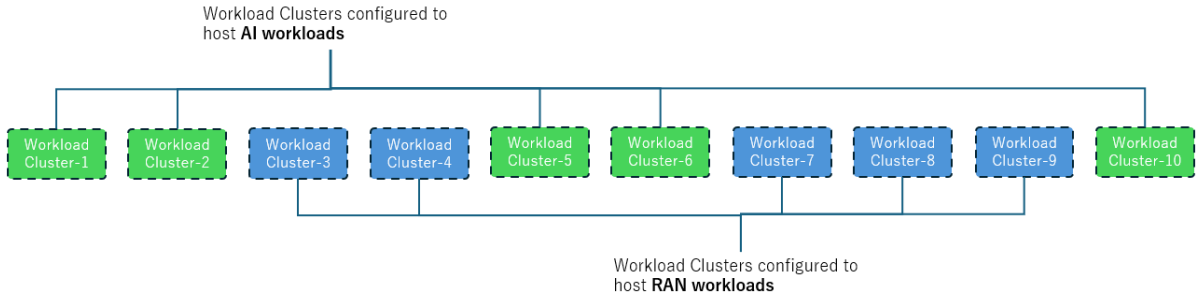

Figure 3: An example of Scenario C

None of the resource management scenarios is excluded from the scope of this document. Note also that these management scenarios are not necessarily mutually exclusive; for example, Scenario A, Scenario B, and Scenario C may be applied simultaneously within the same IT infrastructure. Note also that the arrangement of these scenarios can be dynamically changed without requiring any underlying hardware change.

2.3. RAN Workloads

The definition of "RAN workloads" in the context of AI-and-RAN is virtualized RAN (Radio Access Networks) workloads. Virtualized RAN (vRAN) workloads are containerized and run as CNF (Cloud-native Network Function). The vRAN may or may not be fully compliant with the O-RAN standards.

2.4. AI Workloads

Table 1 defines the different types of AI workloads. It describes each workload type, including whether it is subject to lifecycle management by the AI-RAN P&I Orchestrator. Lifecycle management (LCM) of a workload refers to operations throughout its lifecycle, including creation (onboarding), updating (e.g., applying configuration changes), scaling out, scaling in, scaling up, scaling down, and deletion (decommissioning).

Table 1: Definition of different types of AI workloads

Type	Description	Subject to LCM by the AI-RAN P&I Orchestrator
AI-on-RAN (Type A)	• AI workload developed by a third-party, unrelated to RAN • Tightly integrated with AI Orchestrator • Lifecycle managed by AI Orchestrator	No
AI-on-RAN (Type B)	• AI workload developed by a third-party, unrelated to RAN • Independent from any AI Orchestrator • Lifecycle not managed by AI Orchestrator	Yes
AI-for-RAN (Type A)	• Workload that leverages AI to enhance RAN performance • Tightly integrated with RAN management and orchestration functions • Lifecycle managed by RAN orchestrator	No
AI-for-RAN (Type-B)	• Workload that leverages AI to enhance RAN performance • Loosely coupled with the RAN management and orchestration function • Lifecycle not managed by RAN orchestrator	Yes, if the onboarding request of the workload is directed to the AI-RAN P&I Orchestrator

2.5. AI Accelerators

In the edge data centers, specialized hardware known as “AI Accelerators” is increasingly employed to enhance the performance of machine learning (training) and inference tasks. These accelerators include various processing units, such as GPUs (Graphics Processing Units), TPUs (Tensor Processing Units), and NPUs (Neural Processing Units). Each component is optimized to efficiently handle the computational demands of AI workloads, particularly the large-scale matrix operations and parallel processing typical of deep learning models. In this document, the term AI Accelerator does not exclusively refer to GPUs, but also encompasses TPUs, NPUs, and other processing units that may be developed and become available in the future.

3. Related Work

This section outlines state-of-the-art orchestration technologies from standardization bodies as well as open-source communities, highlighting their capabilities from two perspectives: (a) resource management and (b) lifecycle management of application workloads. This section also outlines prior work related to orchestration, particularly at the container platform (Kubernetes) layer.

3.1. ETSI MANO NFVO

The NFV Orchestrator (NFVO) in ETSI MANO [2] is responsible for orchestration from two main perspectives: (a) resource management, where it coordinates with VIMs to allocate and optimize compute, storage and network resources across infrastructures; and (b) oversees the deployment, scaling, updating, migration, and termination of VNFs and network services by coordinating with VNF Managers, ensuring that multiple VNFs operate together as intended within service policies.

3.2. O-RAN SMO

The Service Management and Orchestration (SMO) framework, as defined in the O-RAN architecture, is designed to orchestrate RAN workloads, including RT/Non-RT RIC, CU, DUs, and RUs. From a resource management perspective, the SMO, in coordination with the O-Cloud and the Infrastructure Management System (IMS), allocates and optimizes the compute, storage, and networking resources required for RAN workloads. From the perspective of lifecycle management for application workloads, the SMO provides a framework in which dedicated services handle onboarding, deployment, configuration, scaling, updating, and termination of RAN functions, while supporting policy-based automation and closed-loop control in collaboration with rApps and xApps.

3.3. Kubernetes and Open Cluster Management

Kubernetes is a container platform that groups multiple servers into a logical unit called a cluster. It enables efficient utilization of computing resources within the cluster and ensures the stable operation of containerized applications. In Kubernetes, the scheduler plays a central role by determining the placement of container applications based on the status and availability of computing resources across the cluster.

Open Cluster Management (OCM), on the other hand, is a platform designed to manage multiple Kubernetes clusters in a unified manner. With OCM, administrators can apply and enforce policies consistently across managed clusters. In addition, OCM provides foundational capabilities—such as observability, policy management, and cluster governance—that support the realization of FCAPS (Fault, Configuration, Accounting, Performance, and Security) functionalities in multi-cluster environments.

3.4. Nephio

Nephio is an open-source initiative under the Linux Foundation's LF Networking program, originally initiated by Google Cloud and supported by major telecom operators and vendors. Its community focuses on enabling intent-driven, cloud-native automation for telecom networks. Nephio excels in treating multiple Kubernetes clusters as managed entities and orchestrating them declaratively through a GitOps-based model.

3.5. ONAP

ONAP, an open-source project of the Linux Foundation's LF Networking program, aims to deliver comprehensive network and service orchestration by automating the lifecycle of physical, virtual, and containerized network functions across distributed cloud environments.

4. Problem Statements

The AI-RAN IT Infrastructure, by definition, is a collection of computing resources designed to host RAN systems alongside AI workloads. In practice, this means that the AI-RAN data centers are deployed across geographically distributed locations, resulting in an infrastructure composed of multiple clusters.

Kubernetes is the de facto standard container platform and serves as the foundation for container-based cloud computing environments. It enables orchestration within a cluster, optimizing computing resource utilization. However, each Kubernetes cluster operates independently and lacks native mechanisms for coordinating across multiple clusters.

SMO in the O-RAN standard primarily focuses on RAN workload management. In AI-RAN, it is essential to manage not only RAN workloads but also AI workloads, which may have different requirements from those of RAN workloads.

Given these factors, there is a need for an entity that provides a holistic view of the entire AI-RAN IT Infrastructure, which consists of multiple Kubernetes clusters, and coordinates the efficient utilization of computing resources such as CPUs and AI accelerators across the entire system.

5. Use Cases

5.1. Use Case 1: Intelligent and Constraint-Aware Workload Placement

The orchestrator determines the target cluster and, optionally, the target node(s) for deploying a given workload, taking various constraints into account. The most basic constraint is the availability of computing resources. That is, the orchestrator shall find a destination cluster/node where required computing resources (CPU, AI accelerator, memory, storage, etc.) are available. Besides, there are other factors that may affect orchestration, such as type, priority, and locality of the workload.

5.2. Use Case 2: Dynamic Resource Allocation and Reconfiguration

The orchestrator dynamically adapts the configuration to the available computing resources based on the orchestration outcome to maximize resource utilization. To be more specific, resource utilization in this context means the use of computing resources such as CPU, AI accelerators, memory, and storage. The higher the resource utilization, the higher the ROI. Achieving high resource utilization requires fine-grained resource management along with dynamic allocation and control mechanisms.

5.3. Use Case 3: SLA-driven Resource Preemption and Prioritization

In shared compute environments, critical workloads such as RAN functions must be prioritized over lower-priority tasks. If a high-priority RAN service request arrives during an ongoing AI job (e.g., model training), the orchestrator pauses or preempts the AI workload and reallocates the resources without violating service-level agreements.

5.4. Use Case 4: External AI Workload Execution

The orchestrator facilitates the exposure of specific parts of the AI-RAN infrastructure to external entities, enabling the execution of AI workloads from third parties. Such capability helps achieve higher resource utilization when the demand for running workloads is limited. The computational demands of RAN workloads are relatively static, whereas those of AI workloads can vary dynamically.

5.5. Use Case 5: Predictive and Proactive Orchestration

Leveraging historical trends and real-time telemetry, the orchestrator employs AI-driven predictive analytics to anticipate workload surges – such as elevated RAN traffic during peak periods or spikes in AI accelerator demand for batch AI inference. This foresight enables proactive resource allocation and dynamic workload shifting, minimizing contention and enhancing system responsiveness and efficiency. RAN traffic patterns are closely correlated with human activity and exhibit well-known 24/7 trends. Based on these patterns, it is possible to predict RAN traffic fluctuations with reasonable accuracy. The AI-RAN P&I Orchestrator may actively leverage this predictive information to orchestrate the AI-RAN IT infrastructure, maximizing the utilization of available computing resources.

6. Requirements

To support intelligent and SLA-compliant orchestration across distributed AI-RAN clusters, the orchestrator shall possess the following key capabilities.

6.1. Functional requirements

6.1.1. Discovering clusters and maintaining an inventory

The Orchestrator shall be able to discover any workload cluster that newly joins the AI-RAN IT Infrastructure. It shall also maintain an up-to-date inventory of all workload clusters that make up the AI-RAN IT Infrastructure.

6.1.2. Monitoring computing resources of the AI-RAN Cloud Infrastructure

The Orchestrator shall be able to collect metrics from each compute node in each workload cluster. The metrics collected are stored in a persistent storage (Database).

6.1.3. Handling requests from external entities to deploy workloads on the AI-RAN Cloud Infrastructure

The Orchestrator shall be able to handle requests from external entities, specifically Consumers (Requesters) and Prediction through the North-Bound Interface.

6.1.4. Identifying candidate compute (e.g., Kubernetes worker nodes) that have sufficient computing resources requested

The Orchestrator shall be capable of identifying candidate compute infrastructure (e.g., Kubernetes worker nodes) that have sufficient computing resources to meet the workload's requirements to be onboarded. The requested resources can be found in the workload manifest.

6.1.5. Dispatching workloads to the destination cluster/node

The Orchestrator shall be capable of dispatching a workload to the target cluster or node. The Orchestrator may take either a declarative approach or an imperative approach to apply the corresponding change to the target cluster.

6.1.6. Conducting priority-aware scheduling of workloads

The Orchestrator shall be capable of taking priority into account in its scheduling. Priority is a type of workload attribute that indicates the relative importance of a workload. For example, RAN workloads (such as vCU and vDU) can be assigned a higher priority than AI-on-RAN workloads.

6.1.7. Conducting data-aware scheduling for AI workloads

Some types of AI workloads require a large amount of training data. For example, an AI-for-RAN workload designed to improve RAN performance requires logs and metrics generated by DUs, resulting in a large data volume. The Orchestrator shall be capable of accounting for data proximity for a given workload in its workload scheduling.

6.1.8. Changing the configuration of the compute (e.g., Kubernetes worker node) according to the demand for running AI or RAN workloads

The Orchestrator shall be capable of dynamically changing the configuration of compute resources (i.e., Kubernetes worker nodes). Certain types of RAN workloads, especially vDU, may impose specific requirements that necessitate OS-level configuration changes (e.g., kernel parameter tuning).

6.1.9. Conducting tenant-aware resource management and orchestration

The Orchestrator shall be capable of managing resources with multi-tenancy awareness. The AI-RAN IT Infrastructure may host workloads from multiple tenants. For example, the Orchestrator shall be capable of enforcing resource quotas per tenant across workload clusters.

6.1.10. Sharing the availability of computing resources with a trusted party

The Orchestrator shall be capable of sharing the availability of computing resources with a trusted party, such as a Domain Specific Orchestrator acting as a broker. This enables the AI-RAN IT Infrastructure to capture and serve external demands for executing AI workloads.

6.1.11. Receiving requests for onboarding AI workloads from external entities through the trusted party

The Orchestrator shall be capable of handling onboarding AI workloads from external entities via a trusted party. The Orchestrator may be involved in request handling either directly or indirectly. In the latter case, an onboarding request for an AI workload may be routed directly to the target cluster shared with the trusted party.

6.1.12. Handling the prediction of computing resource demand from the perspective of RAN and AI, respectively

The Orchestrator shall be capable of interacting with a Prediction function through an East/Westbound Interface. For example, a Prediction function for the RAN domain may provide forecasts of RAN network usage. Such prediction information is valuable and essential for the Orchestrator to efficiently manage resources.

6.1.13. Authenticating and authorizing clients who send requests through a Northbound Interface or any other external entities through the exposed API

The Orchestrator shall be capable of authenticating a client that submits a request through a Northbound Interface. The Orchestrator shall further authorize the request based on the Role-Based Access Control (RBAC) information stored in the persistent data store.

6.1.14. Securing data integrity, data confidentiality, replay protection

The Orchestrator shall apply appropriate security measures, namely SSL/TLS, to secure data exchanged with external systems and to ensure data integrity, confidentiality, and replay protection. The Orchestrator exchanges information with external components through its interfaces. Its internal functional components also exchange information among themselves. The same security principle shall apply to all such data exchanges.

6.1.15. Preempting an on-going job (e.g., an AI training job) and triggering checkpoints for the respective workloads, if supported

The Orchestrator shall be capable of preempting an on-going job (e.g., an AI training job). If the workload supports checkpointing, the Orchestrator shall trigger a checkpoint for the workload subject to preemption before termination.

6.1.16. Handling energy-related metrics (e.g., energy consumption) and taking them into account for scheduling workloads

The Orchestrator shall be capable of handling energy-related metrics. Such metrics include energy consumption per cluster or node, time-series data on energy consumption, energy availability per data center, and information on energy supply and its sources, including whether the energy is renewable.

6.1.17. Exchanging policy information with Domain Specific Orchestrator

The Orchestrator shall be capable of exchanging policy information with a Domain Specific Orchestrator (DSO) to coordinate resource management between the two orchestrators. For example, resource migration may occur from the AI-RAN P&I Orchestrator to the DSO, or vice versa.

6.2. Non-functional requirements

For future study

7. System Architecture

This section provides an overview of the network architecture and the functional components that comprise the AI-RAN P&I Orchestrator. These components are designed based on the requirements outlined in the previous section. In other words, the functional components are structured to satisfy those requirements.

7.1. Network Overview

Figure 5 illustrates an overview of the AI-RAN network. There are two types of clusters: management clusters and workload clusters. The management clusters are primarily designed to host management functions, such as the AI-RAN P&I Orchestrator, git servers, and image registries. The workload clusters, on the other hand, are primarily designed to accommodate workloads—including both AI and RAN workloads. From the perspective of computing resources, all worker nodes across all workload clusters collectively represent the compute resource assets of the AI-RAN IT Infrastructure. In contrast, management clusters contribute mainly through orchestration and auxiliary services.

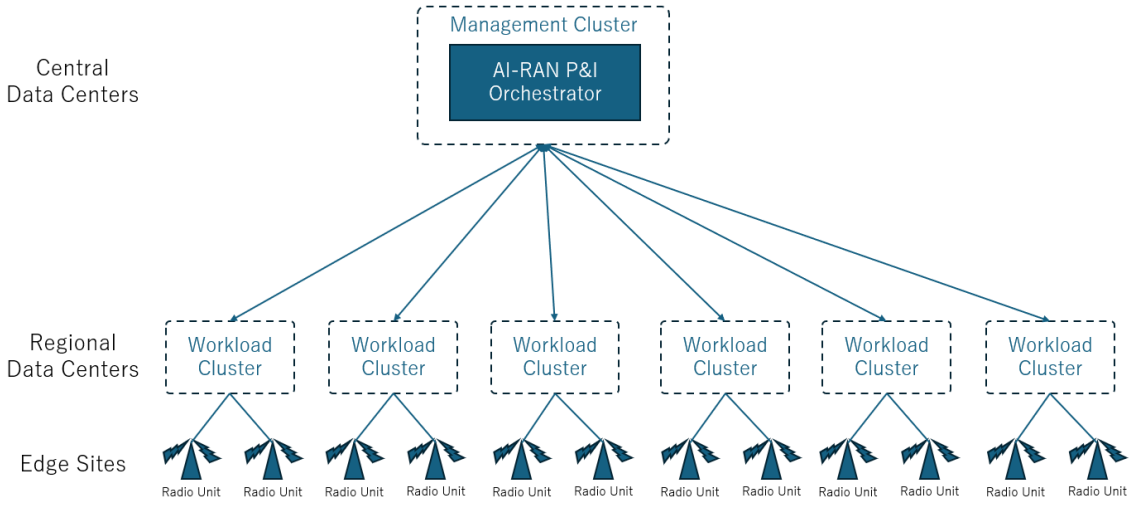

Figure 5: AI-RAN Network Overview from Container Platform perspective

7.2. Functional Components

This section focuses on the AI-RAN P&I Orchestrator itself. Figure 6 illustrates its system architecture. The orchestrator is composed of five functional components:

1. Multi-Cluster Management Function
2. Metrics Collection Function
3. Multi-Cluster Scheduling Function
4. Dynamic Scoring Function
5. Resource Arrangement Function

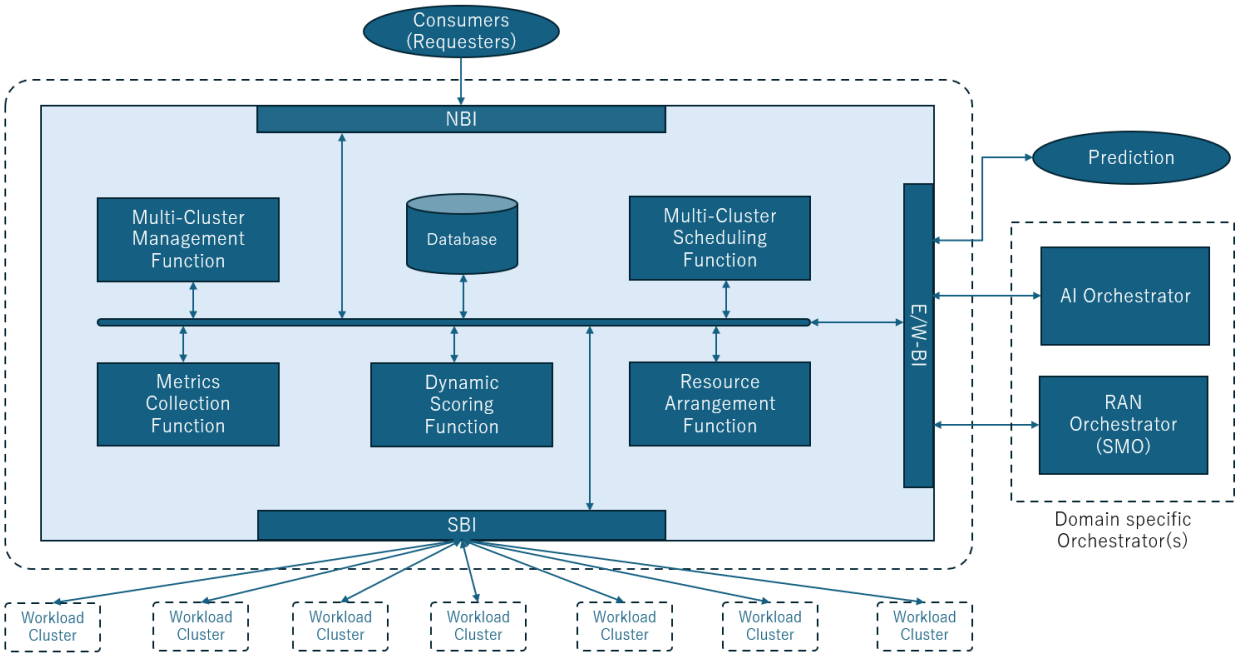

Figure 6: System architecture of AI-RAN P&I Orchestrator

The **Multi-Cluster Management Function** is responsible for managing multiple clusters—particularly the workload clusters—in an integrated manner. Specifically, it establishes associations between the management cluster and each workload cluster and creates and maintains an inventory of the workload clusters. When a new workload cluster joins the AI-RAN IT Infrastructure, the function detects the cluster and updates the inventory accordingly. The inventory file is stored in persistent storage (Database).

The **Metrics Collection Function** collects metrics from all clusters listed in the workload cluster inventory. Specifically, it retrieves metrics by accessing each node's metrics endpoints within the workload clusters. The implementation details of the metrics collection process may vary; for example, the function may gather metrics by accessing endpoints exposed by each worker node in every workload cluster. The collected metrics are stored in persistent storage (Database).

The **Multi-Cluster Scheduling Function** schedules workloads across a multi-cluster environment. This function takes the following inputs:

- A workload manifest submitted through the NBI
- Dynamic scoring results retrieved from the Database

Using this information, the function schedules workloads across multiple clusters. Based on the dynamic scoring results, it determines the desired state of resources within the AI-RAN IT Infrastructure. A simple example of a desired state is deploying a given workload to a specific cluster. A more complex example involves deploying a workload (e.g., AI-on-RAN workload) to

a cluster where a worker node must be reconfigured—e.g., switching its role from RAN to AI—to accommodate the new workload.

The **Dynamic Scoring Function** evaluates and scores multiple potential execution targets for a given workload with specific attributes, based on corresponding evaluation criteria, to assess the expected outcome when the workload is executed on each target. The function takes the following inputs:

- evaluation criteria,
- metrics collected from workload clusters
- information on workload clusters

The output is the evaluation result (score) of each resource arrangement.

The **Resource Arrangement Function** applies intended resource changes to target cluster(s) or node(s) via the Southbound Interface (SBI). Specifically, it receives input from the Multi-Cluster Scheduling Function as an array of operations that describe the desired resource changes. These operations may include:

- deploying workloads (e.g., Kubernetes Deployments), and
- modifying node configurations.

The exact method for applying resource changes is implementation-specific. For example, one may adopt a declarative approach with GitOps or an imperative approach via the Kubernetes API. In the declarative case, the Resource Arrangement Function generates manifest files for the resources to be modified and pushes them to a Git repository.

7.3. Orchestrator Processes

In this section, concrete procedural steps are presented so that readers can gain a clearer understanding of the roles of each functional component within the AI-RAN P&I Orchestrator. A typical use case, the processing of an application onboarding request, is used as an example.

Assumptions:

- The Application to be onboarded is an AI workload, specifically an AI-on-RAN type B application.
- The AI workload has no locality requirement; in other words, the Consumer does not care which workload cluster executes the workload.
- The AI-RAN administrator aims to minimize the overall energy consumption of the AI-RAN IT Infrastructure.

- The workload clusters are geographically distributed, and the energy supply conditions differ across data centers. More specifically, renewable energy availability varies across data centers.

Here are the procedural steps taken in this scenario:

1. A Consumer submits an onboarding request for an AI workload. The request is expected to include the following information: the workload manifest (e.g., Kubernetes Deployment manifests) and workload attributes.
2. The NBI authenticates the Consumer and authorizes the request.
3. The NBI enqueues the request to the Job Request Queue (JRQ), which is stored in a persistent data store (Database). The JRQ is continuously monitored by the Multi-Cluster Scheduling Function (MCSF) and the Dynamic Scoring Function (DSF).T
4. The Dynamic Scoring Function (DSF) is triggered to perform scoring for each workload cluster, considering factors such as real-time resource availability (CPU, memory, and AI Accelerators) and renewable energy availability. The DSF stores the resulting scores in the Database.
5. Once the dynamic scoring results become available, the Multi-Cluster Scheduling Function (MCSF) is triggered. The MCSF determines the desired state of the AI-RAN IT Infrastructure – in this case, the optimal cluster and node to execute the requested workload – by selecting a cluster that provides sufficient computing resources and an economical energy supply.
6. The Resource Arrangement Function (RAF) derives the detailed configuration required to transform the AI-RAN IT Infrastructure in into the desired state. In this case, the necessary changes involve deploying the AI workload on the selected cluster/node. Specifically, the RAF adjusts the deployment manifest by specifying the designated node using appropriate node affinity settings (such as nodeSelector or nodeAffinity). The RAF not only prepares the required manifest but also orchestrates the processing order as needed.

8. AI-RAN P&I Orchestrator Interfaces

The AI-RAN P&I Orchestrator interacts with various external components through dedicated APIs. As illustrated in Figure 6, it exposes three types of interfaces:

- North Bound Interface (NBI)
- East/West Bound Interface (E/W-BI)
- South Bound Interface (SBI)

8.1. North Bound Interface

The Northbound Interface (NBI) enables the orchestrator to receive requests from Consumers (also referred to as Requesters). A typical Consumer is an entity that intends to deploy and run AI-on-RAN workloads within the AI-RAN IT Infrastructure. Consumers may submit workload-onboarding requests that include a workload manifest. These requests can optionally include annotated metadata, referred to as Workload Attributes. It is assumed that the NBI inherently provides authentication and authorization capabilities for API requests.

8.2. East/West Bound Interface

The East/Westbound Interface (E/W-BI) enables the orchestrator to communicate with Domain-Specific Orchestrators (DSOs). A Domain-Specific Orchestrator refers to an orchestration function tailored to a particular domain, such as RAN or AI/ML. For example, in an O-RAN-compliant RAN system, the SMO (Service Management and Orchestration) can be considered a DSO for the RAN domain.

Another type of entity that may interact with the E/W-BI is a Prediction component. This entity performs forecasting of future events or conditions within a specific domain. For example, a RAN Prediction function forecasts radio access network (RAN) usage patterns. Since RAN usage is closely correlated with human activity, it often exhibits well-known daily trends over a 24/7 cycle. Another example is an energy-related Prediction function, which may provide short-term forecasts of renewable energy availability in specific regions.

8.3. South Bound Interface

The Southbound Interface (SBI) enables the orchestrator to communicate with workload clusters across multiple functional aspects.

- First, the Multi-Cluster Management Function establishes associations with workload clusters. This function and its agents—deployed within each workload cluster—communicate bidirectionally via the SBI.
- Second, the Metrics Collection Function retrieves metrics from each cluster through the SBI.

- Third, the Resource Arrangement Function applies desired resource changes to the target cluster(s) or node(s) via the same interface.

8.4. Role Separation from Domain Specific Orchestrator

Table 2 defines the respective roles of the AI-RAN P&I Orchestrator and the Domain-Specific Orchestrator to clarify the separation of responsibilities between them. Resource management includes monitoring the utilization of computing resources through metrics collection, and arranging resources for specific computational purposes (e.g., reconfiguring nodes from AI to RAN, or vice versa).

Table 2: Roles of AI-RAN P&I Orchestrator and Domain Specific Orchestrator

	AI-RAN P&I Orchestrator	Domain Specific Orchestrator (DSO)
Resource Management	• Has visibility of the entire AI-RAN IT Infra • Has precedence in changing the domain boundaries	• Has visibility of all the resources under the control of the DSO
Lifecycle Management	• Manages the lifecycle of workloads whose LCM request is directed to the AI-RAN P&I Orchestrator • Destination clusters and nodes could be the ones that are not bound to any DSO	• Manages the lifecycle of the domain-specific workloads • Destination clusters and nodes are the ones under the DSO's control

9. Further Study

The following items are for further study.

- **Orchestration of hardware, network, and storage:** So far, the focus has been on orchestrating computing resources such as CPUs, memory, and AI accelerators. Considering the broader role of the AI-RAN P&I Orchestrator, its scope should be extended to include the orchestration of hardware, network, and storage resources.
- **Clarify the meaning and role of the "default" domain:** Further clarification is needed regarding the meaning, role, and intended use of the "default" domain within the AI-RAN IT Infrastructure. In addition, the scope of the functions that the AI-RAN P&I Orchestrator can perform within this domain should be clearly defined.
- **Interaction between the AI-RAN P&I Orchestrator and the RAN Orchestrator:** Further study is needed on how the AI-RAN P&I Orchestrator should interact with the RAN Orchestrator (e.g., the O-RAN SMO). This includes identifying the information that must be exchanged and detailing the negotiation process between the two orchestrators.
- **Non-functional orchestrator requirements:** Additional requirements may be defined in future versions of this document.

10. Glossary of Terms

- AI-RAN P&I Orchestrator
- AI-RAN IT Infrastructure
- Lifecycle Management (LCM)
- Management Cluster
- Workload Cluster
- Domain Specific Orchestrator (DSO)
- Workload Attributes
- Virtualized Infrastructure Manager (VIM)
- Return on Investment (ROI)

11. References

[1] AI-RAN Alliance, "AI-RAN Architecture Overview and Component Definitions", Version 1.2, February 2026

[2] ETSI GS NFV-006 "NFV Release 4; Management and Orchestration; Architectural Framework Specification, V4.5.1, 2024-05, https://www.etsi.org/deliver/etsi_gs/NFV/001_099/006/04.05.01_60/gs_nfv006v040501p.pdf

12. Document History

Version	Edited by	Comments	Date Updated
0.1	Shinta Sugimoto (SoftBank Corp.)	Table of contents	2025/04/23
0.2	Shinta Sugimoto (SoftBank Corp.)	Filled in texts for the following sections: 1. Introduction 2. Assumptions and Prerequisites 4. Problem Statements 5. Use cases	2025/05/10
0.3	Shinta Sugimoto (SoftBank Corp.)	Reflected comments provided for rev0.2: Section 2.3 RAN Workloads Section 2.5 AI Accelerators Section 5 Use Cases Section 6 Requirements	2025/07/03
0.4	Shinta Sugimoto (SoftBank Corp.)	Reflected comments provided for rev0.3 Section 2.4 Section 6.1.17 Section 7 (System Architecture) Section 8 (Interfacing with External Systems) Section 9 (Glossary of Terms)	2025/08/13
0.5	Shinta Sugimoto (SoftBank Corp.)	Reflected the outcome of the discussion in the WG meeting	2025/09/08
06	Shinta Sugimoto (SoftBank Corp.)	Reflected the comments and feedback from WG2 members and the results of WG meeting discussions	2025/09/22
07	Shinta Sugimoto (SoftBank Corp.)	Reflected comments and feedback from WG2 members and the results of WG meeting discussions	2025/10/07
08	Shinta Sugimoto (SoftBank Corp.)	Added detailed descriptions of individual requirements in Section 6 (Requirements) Newly introduced Section 7.3 (Processes of Orchestrator)	2025/10/21
09	Shinta Sugimoto (SoftBank Corp.)	Responded to the comments given to versions 06 and 07 Responded to the feedback provided in the WG2 meeting (10/10) Responded to the comments given to version 08	2025/10/29
10-rc1	Shinta Sugimoto (SoftBank Corp.)	Cleaned up the documents (fixed editorial errors) Reflected the texts suggested for Section 2.2 Added section 9 (Further Study)	