Chapter 6

Containerized Privacy Protection in High Performance Computing Clusters: An Application in Health Care

This chapter focuses on a task-based containerization approach, particularly in integrating Docker containers with Federated Learning (FL). This integration streamlines the management of resources, dependencies, and libraries, facilitating the efficient execution of machine-learning tasks. By deploying FL systems across distributed infrastructures, the approach effectively addresses data privacy and security concerns, as well as the intricacies of resource allocation and sensitive data handling, making it a powerful solution for data-intensive challenges in the realm of HPC Cluster.

6.1 Introduction

In the intricate landscape of HPC, where machine learning models are leveraged to tackle complex challenges, resource optimization is crucial. The integration of machine

learning within HPC workflows encounters resource limitations, especially when handling extensive datasets. To address this, we turn to containerization, a technology that encapsulates computational environments for consistency and reproducibility across diverse HPC systems. Our focus is on how Docker containers are employed in HPC using a task-based containerization approach, with a specific emphasis on their integration with FL, particularly the FedAvg algorithm. Containerization streamlines the management of resources, dependencies, and libraries, allowing the agile and efficient execution of machine learning tasks. The incorporation of FedAvg underscores the optimization of distributed machine learning workflows within containers. This integration empowers researchers to deploy FL systems seamlessly across distributed infrastructures while meticulously addressing data privacy and security concerns. It also addresses the intricacies of resource allocation and sensitive data handling, making it a powerful solution for data-intensive challenges.

The Conundrum of Machine Learning and Privacy in HPC: The pursuit of scientific discoveries, simulations, and data-driven insights relies heavily on substantial computational resources. Machine learning, particularly deep learning models, thrives on extensive datasets. However, a significant challenge arises when the data required for these models is distributed across various organizations due to concerns regarding data privacy and legal obligations. This challenge is especially conspicuous in sectors like healthcare, where data is not only voluminous but also highly sensitive, subject to a myriad of regulations. In response, privacy-preserving methods like FL have emerged as a solution, allowing the training of machine learning models without the need for centralized data sharing. The distributed nature of FL introduces challenges, such as unbalanced data distribution and the presence of non-IID (Non-Independently and Identically Distributed) data. This work is centered around the development and validation of a FL-based system for the detection of COVID-19 from Chest X-ray images within the HPC landscape. This endeavor is groundbreaking, representing a pioneer-

ing step at the intersection of HPC and healthcare. In the subsequent sections, we delve deeper into the existing body of related work, introduce a federated optimization framework tailored to the challenge of COVID-19 detection, and meticulously dissect the experimental results. Our work serves as a testament to the convergence of HPC, FL, and data privacy in the pursuit of solutions to some of the most pressing global challenges. Currently, multiple diagnostic methods are available for the detection of coronavirus; however, Chest X-ray images and CT scans have emerged as widely accepted standard diagnostic techniques [**?**], [**?**], [**?**]. Given that COVID-19 primarily targets the respiratory tract's epithelial cells, the health of a patient's lungs can be assessed using Chest X-ray images. Furthermore, the ubiquity of X-ray imaging machines in healthcare facilities suggests the potential utility of X-ray images for COVID-19 testing, even in the absence of dedicated test kits. Notably, the application of Artificial Intelligence to analyze chest X-ray images offers a rapid and cost-effective approach to COVID-19 screening. Consequently, numerous research endeavours have been dedicated to predicting the outbreak and diagnosing COVID-19 using machine learning techniques [**?**], [**?**] and [**?**], [**?**]. In our present study, we have directed our focus towards the development and validation of a FL system for the detection of COVID-19 from Chest X-ray images, constituting the fundamental innovation of this article. To the best of our knowledge, this investigation represents the inaugural attempt to apply FL to X-ray images for COVID-19 detection. At its core, our exploration hinges on the integration of containerization, the FedAvg algorithm, and FL within the HPC environment. This integration serves as the linchpin in overcoming resource limitations and ensuring data privacy in this intricate landscape.

FL is a machine-learning approach that allows multiple parties to collaboratively train a shared model while keeping their data decentralized and private. It is particularly useful when data cannot or should not be centralized in a single location due to privacy, security, or regulatory concerns. Let's consider a scenario with N participating

parties (such as devices or organizations) denoted as $P_1, P_2, P_3, P_4, \ldots, P_N$. Each party P_i has its own local dataset D_i, and the goal is to train a global machine learning model M that generalizes from the collective data of all parties without sharing their individual datasets. In FL, each party P_i, initializes its local model parameters Θ_i. FL proceeds in a series of communication rounds. In each round $t = \{1, 2, \ldots, t\}$, each party P_i, computes an update $\Delta\theta_{it}$, to its local model parameters based on its local dataset D_i. This update is typically done by minimizing a local loss function with respect to Θ_i. The updates $\Delta\theta_{it}$ from all parties are aggregated to create a global update $\Delta\theta_t$, which is a weighted average or some other aggregation function. The global update $\Delta\theta_t$ is then applied to the global model parameters to obtain a new global model Θ_t. The global update $\Delta\Theta_t$ is communicated to all parties. FL continues for a fixed number of communication rounds (T) or until convergence criteria are met.

6.2 Related work and major contribution

Fuelled by recent advancements in FL and IoT, numerous literature reviews on this subject have emerged. For instance, a comprehensive survey of the FL concept and its essential system components, including data distribution, machine learning models, privacy mechanisms, and communication architecture, was conducted in a study referenced as [**?**]. Similarly, in [**?**], the authors explored the FL concept from an architectural standpoint and delved into its fundamental applications in the realm of business. Furthermore, additional articles, as denoted by [**?**] [**?**] [**?**], elucidated FL architectures, software, platforms, and protocols, while also delving into potential research challenges inherent in FL implementations. Concurrently, the research presented in [**?**] delved into the technical hurdles encountered within FL systems, encompassing issues such as data privacy bottlenecks and network vulnerabilities. Further exploration is warranted to comprehensively explore the potential benefits of FL in the Internet of Medical Things (IoMT), considering the rapid growth of recent research in this field [**?**].

On the other hand, Docker serves as a container virtualization platform. Unlike virtual machines (VMs), which abstract physical hardware, containers provide an abstraction at the application layer. Consequently, they simplify the deployment and management of applications by offering a consistent execution environment and interface [**?**]. In recent times, a significant body of research has emerged, focusing on the development of deep learning algorithms for the detection of diseases using chest X-ray images [**?**]. Notably, a pioneering effort in [**?**] introduced an algorithm capable of pneumonia detection from chest X-ray images, surpassing the diagnostic accuracy of practising radiologists through the use of a Dense Convolutional Network. Xu et al. [**?**] employed a hierarchical Convolutional Neural Network (CNN) to classify X-ray images into normal and abnormal categories. Additionally, a descriptive study [**?**] involving radiology images from COVID-19 cases revealed their potential for diagnostic purposes and early disease recognition. Consequently, numerous studies have emerged, proposing various techniques for COVID-19 detection based on radiology images. Hemdan et al. [**?**] and Wang and Wong [**?**] harnessed deep learning models for COVID-19 diagnosis using chest X-ray images. Nour and Cömert [**?**] utilized a CNN model to extract discriminative features from X-ray images, subsequently applying them to three machine learning algorithms: k-nearest neighbour, support vector machine, and decision tree. Gupta et al. [**?**] introduced an integrated stacked deep convolutional network for COVID-19 and pneumonia detection in chest X-ray images by identifying abnormalities. Zhang et al. [**?**] developed a deep learning-based model with high sensitivity to detect COVID-19 in chest X-ray images, enabling swift and reliable screening. In [**?**] , a deep model for early COVID-19 detection using X-ray images was introduced, achieving high accuracy for binary and multi-class scenarios. Narin et al. [**?**] and Chowdhury et al. [**?**] conducted training and comparative analysis of multiple pre-trained CNN-based models to detect COVID-19 in chest X-ray images. More recently, Demir [**?**] proposed a deep Long short-term memory (LSTM) architecture to automatically identify COVID-19 cases

from X-ray images, learned from scratch. Other research works [**?**] [**?**], [**?**] [**?**], have focused on detecting COVID-19 positive cases from chest CT scans using CNN-based models. However, a limitation of these centralized models arises from the reluctance of medical institutions to compromise doctor-patient confidentiality by sharing sensitive medical images like X-rays for training. In response, the healthcare research community has increasingly recognized the potential of FL as an effective means to facilitate knowledge sharing among medical institutions while ensuring privacy. Lee et al. [**?**], for instance, presented a privacy-preserving platform within a federated setting, enabling patient similarity learning across institutions without the need to share patient-level information. Similarly, Huang et al. [**?**] addressed the challenge of non-IID ICU patient data by clustering patients into clinically meaningful communities and optimizing predictive performance for mortality and ICU stay time. More recently, Baheti et al. [**?**] employed the concept of FL for the detection of pulmonary lung nodules in CT scans, highlighting the promise of FL in the healthcare domain.

In this work, we introduce a groundbreaking approach that amalgamates three key elements: FL, HPC clusters, and Docker Swarm. Our primary objective is to predict COVID-19 from chest radiology images, aiming to significantly enhance healthcare practices and pandemic response capabilities. In this chapter, we delve into the intricacies of our approach, elucidate our unique contributions to the field, and underscore the crucial intersection of FL with the IoMT. Recent advances in technology have paved the way for substantial progress in both FL and the IoT. Comprehensive reviews and studies have explored the FL concept, its constituent system components, and its applications across various domains. Similarly, the IoT has garnered considerable attention for its potential to revolutionize various industries, particularly healthcare. However, our research represents a pioneering effort to bridge these domains, particularly in the context of combating COVID-19.

6.3 Overview of the approach and HPC environment

We have successfully implemented FL, a decentralized machine learning paradigm, to collaboratively train models while preserving the privacy of distributed data. PARAM Smriti, (High Performance Computing), enabling us to achieve precise and efficient COVID-19 prediction from chest radiology images. To streamline our research pipeline, we've adopted a task-based container approach where each research task has been meticulously encapsulated within a dedicated Docker container. These tasks, including data preprocessing, model training, model evaluation, and report generation, have been seamlessly integrated. In Task 1, the Data Preprocessing Container, we executed data preprocessing by utilizing a Dockerfile specifying a base image (e.g., Python) and installing essential libraries such as Pandas, NumPy, tensorflow-federated, and OpenCV. Our Python script for data preprocessing was incorporated within the container, and the Dockerfile's entry point was configured to execute the preprocessing script. Task 2, the Model Training Container, efficiently conducted the model training process. The Dockerfile specified a base image equipped with TensorFlow and other essential machine learning libraries. The model training code, implemented as a Python script, was housed within the container, and the Dockerfile's entry point was established to execute the training script. In Task 3, the Model Evaluation Container, we conducted model evaluation, producing valuable insights. The Dockerfile incorporated the required evaluation libraries, and our model evaluation code was embedded within a Python script housed in the container. The Dockerfile's entry point executed the evaluation script seamlessly. Task 4, the Reporting Container, effectively generated insightful reports and visualizations. The Dockerfile specified a base image enriched with reporting libraries, including Matplotlib. Our code for generating visualizations and reports was integrated into a Python script within the container, and the Dockerfile's entry point adeptly executed the reporting script. In this approach we have demonstrated its efficiency and ease of management, contributing to enhanced reproducibility and main-

tenance. The orchestration of these containers within our HPC cluster environment has been accomplished using Docker Swarm. This orchestration ensures scalability and facilitates the management of FL components. Using Docker Compose, we've effectively defined how these containers interact, specifying task dependencies and orchestrating their execution flow. This comprehensive setup has enabled us to harness the power of FL while efficiently managing our COVID-19 prediction research pipeline. This successful integration positions us at the forefront of COVID-19 prediction and pandemic response using the Federated Averaging (FedAvg) approach. In the rest of this section, we introduce PARAM Smriti where we implemented our solution.

PARAM Smriti: The PARAM Smriti HPC Facility is a cutting-edge facility established as part of the National Supercomputing Mission's (NSM) build approach, boasting an impressive peak computing power of 838 TFLOPS. This state-of-the-art system was meticulously crafted and deployed by the HPC Technologies team at the Centre for Development of Advanced Computing (C-DAC), utilizing a heterogeneous and hybrid configuration. This configuration combines Intel Xeon Cascade Lake processors with NVIDIA Tesla V100 GPUs. The PARAM Smriti HPC system caters to a wide array of scientific domains, making it a versatile resource.

It finds applications in Agricultural Biotechnology, Food and Nutritional Biotechnology, Computational Fluid Dynamics, Computational Chemistry, Artificial Intelligence, Computational Physics, Manufacturing Process Modeling, Computational Biology, Data Science, Weather & Climate, Oil & Gas, Seismic, Life Sciences, and Material Sciences. The system's infrastructure includes 2 Master nodes, 4 Login nodes, 4 Service nodes, and a substantial 156 compute nodes, which encompass both CPUs and GPUs. This comprehensive setup culminates in a remarkable total peak computing capacity of 838 TFLOPS. Fig. 6.1 and Fig. 6.2 illustrate PARAM smriti and its architecture, respectively. The technical specification of PARAM Smriti is illustrated in Figure. 6.3.

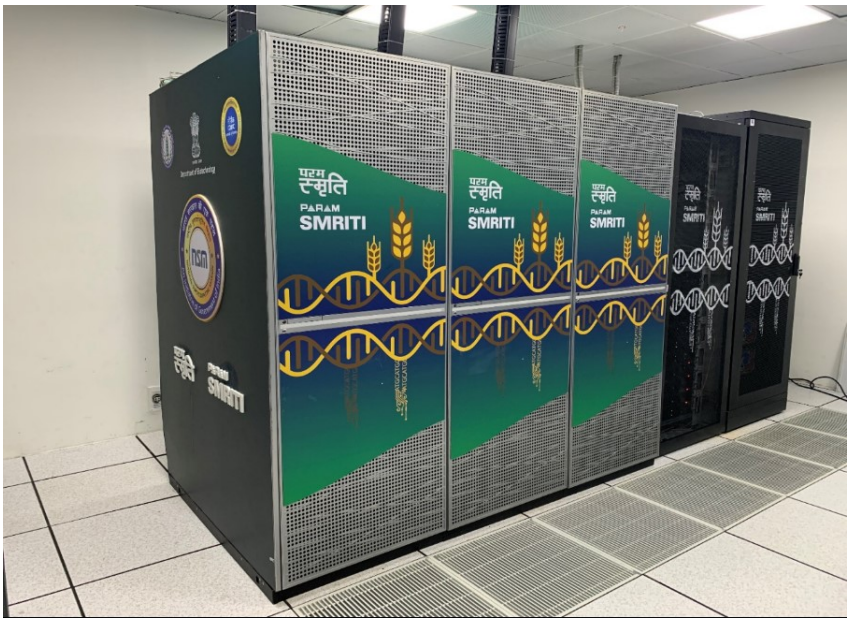


Figure 6.1: PARAM Smriti

Job Scheduler in PARAM Smriti: The PARAM Smriti HPC features the installation of a Slurm job scheduling system. Slurm, short for Simple Linux Utility for Resource Management, serves as a robust workload manager. It offers a structured framework for managing job queues, efficiently allocating compute nodes, and initiating and supervising job executions.

Parameters used in SLURM job script: Job flags in the context of Slurm are employed alongside the SBATCH command. To integrate a SLURM directive into a script, the prescribed syntax is `#SBATCH <flag>`. Certain flags are also utilized in conjunction with the srun and salloc commands. Table 6.1 illustrates the SLURM Flags and description.

6.4 System model

We consider the objective function, denoted by $F(w)$, to represent the loss incurred when using a model with parameter vector w to make predictions on a dataset. It quantifies how well the model fits the data. In the context of FL, data points are distributed across multiple clients, where D_i represents the set of data points on client

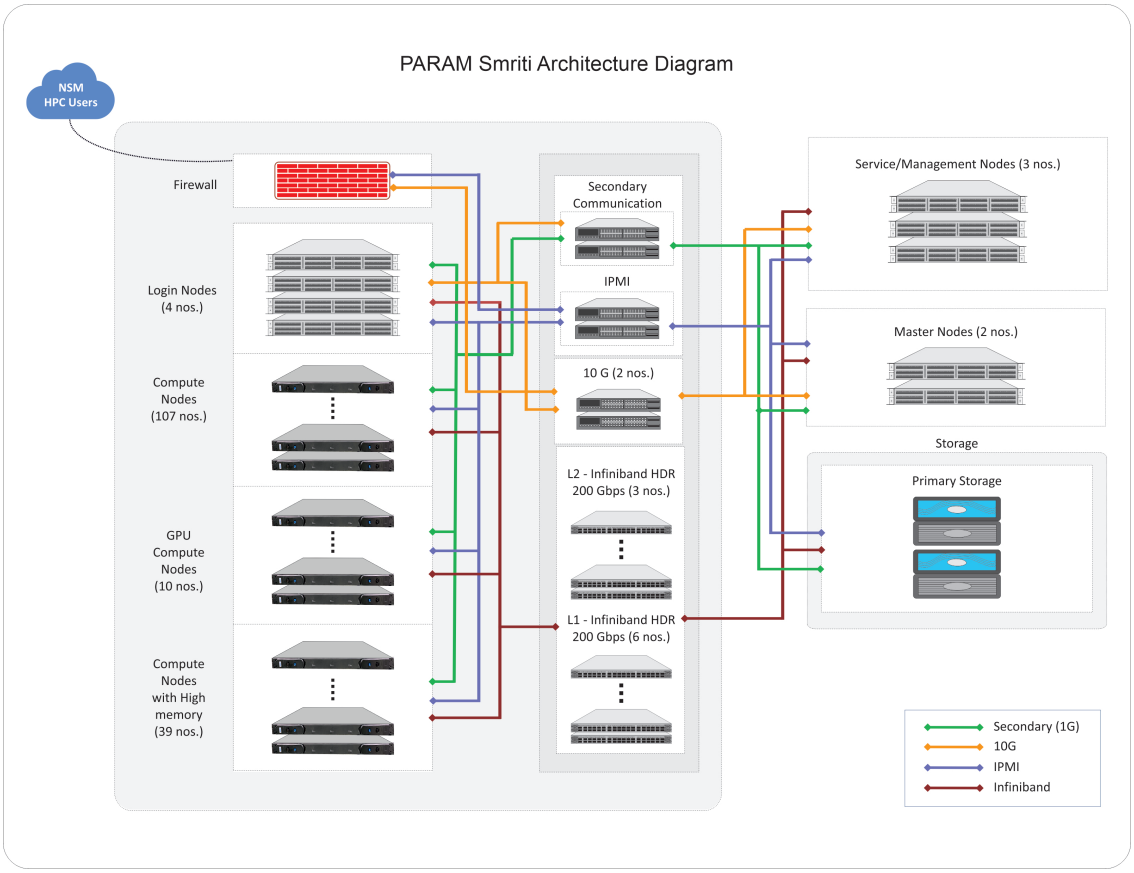


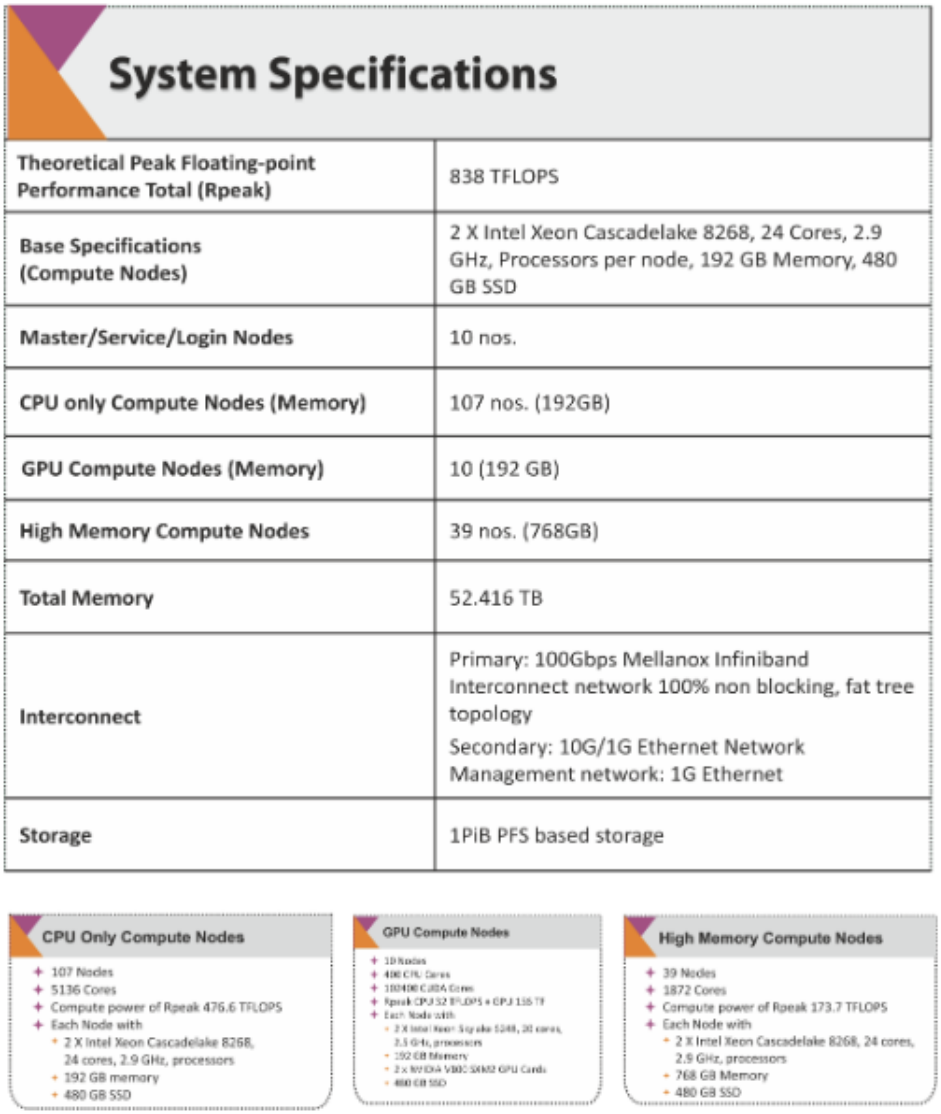
Figure 6.2: PARAM Smriti architecture diagram.

i, and N_i is the number of data points on client i.

The optimization objective in FL is to minimize the global loss, which is defined as the sum of losses across all clients. This is represented by the following equation:

$$\min(w) = F(w) = \frac{1}{n_k} \sum_{x \in D_i}^{N} f_i(w) f(w, x)$$

In FedAvg, clients synchronize their updates with a central server. This means that clients train their local models and send updates to the server in rounds, and the server aggregates these updates to obtain a global model. When data is independently and identically distributed (IID) across clients, FedAvg achieves convergence results comparable to a central model. This means that if each client's data is representative of the global distribution, FedAvg performs well. We have a FL architecture with four Docker



System Specifications

Theoretical Peak Floating-point Performance Total (Rpeak)	838 TFLOPS
Base Specifications (Compute Nodes)	2 X Intel Xeon Cascadelake 8268, 24 Cores, 2.9 GHz, Processors per node, 192 GB Memory, 480 GB SSD
Master/Service/Login Nodes	10 nos.
CPU only Compute Nodes (Memory)	107 nos. (192GB)
GPU Compute Nodes (Memory)	10 (192 GB)
High Memory Compute Nodes	39 nos. (768GB)
Total Memory	52.416 TB
Interconnect	Primary: 100Gbps Mellanox Infiniband Interconnect network 100% non blocking, fat tree topology Secondary: 10G/1G Ethernet Network Management network: 1G Ethernet
Storage	1PiB PFS based storage

Figure 6.3: Illustration of PARAM Smriti technical specification.

containers acting as clients and an HPC (High-Performance Computing) cluster serving as the central server. The goal is to train a machine learning model without sharing raw data but only model updates. Fig. 6.4 illustrates the generated architecture of the proposed approach. Here's how the architecture works:

Step 1 (Initialization): Initially, a global machine learning model is created and hosted on the central server. All client containers receive a copy of this initial model.

Step 2 (Local Training): Each client independently trains its local model using its private data. This training can be done using a variety of machine learning algorithms,

Table 6.1: SLURM Flags and Descriptions

Resource	Flag Syntax	Description
partition	–partition=partition name	Partition is a queue for jobs.
time	–time=01:00:00	Time limit for the job.
nodes	–nodes=2	Number of compute nodes for the job.
cpus/cores	–ntasks-per-node=8	Corresponds to the number of cores on the compute node.
resource feature	–gres=gpu:2	Request use of GPUs on compute nodes.
account	–account=group-slurm-account	Users may belong to groups or accounts.
job name	–job-name="lammps"	Name of the job.
output file	–output=lammps.out	Name of the file for stdout.
email address	–mail-user=username@iitbhu.ac.in	User's email address.
access	–exclusive	Exclusive access to compute nodes.

such as deep neural networks, decision trees, or any other suitable model.

Step 3 (Model Updates): After local training, each client computes the model update. This update represents the difference between the local model's parameters before and after training. These updates are typically small and do not contain any raw data. The client containers then send these updates to the central server. This is where Docker containers play a role in packaging and transmitting these updates.

Step 4 (Aggregation0: The central server receives these model updates from all client containers. It aggregates these updates to create an improved global model. Common aggregation methods include simple averaging, weighted averaging, or more sophisticated techniques like secure multi-party computation (SMPC) to ensure data privacy.

Step 5 (Model Deployment): The updated global model, with its respective weights adjusted by aggregating all the client updates, is then deployed back to the client containers. Each client replaces its local model with the updated global model.

Iterative Process: Steps 2 to 5 are iteratively repeated. Each iteration involves local training, computing model updates, aggregating updates at the central server, and deploying the updated global model. The process continues for multiple rounds until the global model converges to a satisfactory level of accuracy.

Architecture Diagram:

General Architecture Diagram:

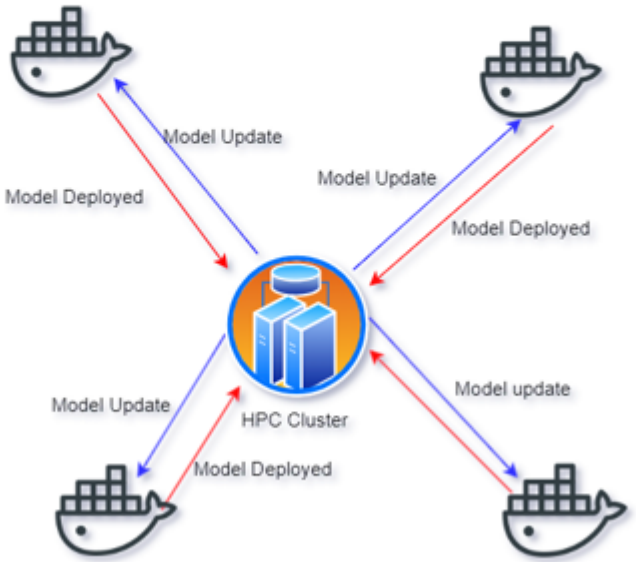


Figure 6.4: General architecture diagram.

The approach uses the following components for running the tasks.

HPC Cluster (Central Server): The HPC Cluster serves as the central server for coordinating FL. It manages model updates and training across the various containers while preserving data privacy.

Data Preprocessing Container (Task 1): This container handles data preprocessing for the FL process. It is responsible for preparing and preprocessing data from different sources while ensuring data privacy.

Model Training Container (Task 2): In this container, machine learning models are trained using FL techniques. The HPC cluster coordinates the model training process, aggregating updates from various sources to improve the global model.

Model Evaluation Container (Task 3): This container evaluates the performance of the

models. The HPC cluster manages the evaluation process, aggregating insights and results from different containers to assess the model's effectiveness.

Reporting Container (Task 4): The reporting container generates reports and visualizations based on the aggregated data. The HPC cluster oversees the reporting process, facilitating insights and visualization generation.

6.5 Task-based containerization approach

In this section, we illustrate the deployment of the model and present task-based containerization approach.

6.5.1 Deployment scenario

Initially, the machine learning model with its initial weights is deployed within the Model Training Container (Task 2). The model architecture and weights are defined in the Python script within this container. During the FL process, the HPC cluster, acting as the central server, orchestrates the model updates. As new data becomes available from various sources, the model is updated by aggregating updates from different containers (e.g., Data Preprocessing, Model Training, Model Evaluation) while preserving data privacy. Model updates with weights are transmitted from the respective containers to the HPC cluster, which serves as the central hub for model aggregation and refinement. FL techniques are employed to ensure data privacy and security during this process. The HPC cluster manages the iterative process of updating the model with new data from the containers to enhance its accuracy and effectiveness in predicting COVID-19 from chest radiology images. This updated architecture diagram and explanation highlight the use of FL in the context of an HPC cluster as the central server for coordinating model updates and deployment while preserving data privacy.

The HPC cluster "PARAM Smriti" functions as the central computing node orchestrating our high-performance computing environment. It delivers an extraordinary

theoretical peak performance of 838 TFLOPS, thanks to its powerful core compute nodes featuring dual Intel Xeon Processors with 24 cores each, clocked at 2.9 GHz, and substantial memory and SSD storage. The cluster also includes specialized nodes such as Master/Service/Login nodes, CPU-only compute nodes, GPU compute nodes, and high-memory compute nodes.

To facilitate seamless communication and data transfer, the cluster is equipped with a primary 100 Gbps IO interconnect and a secondary IOG/IG Ethernet network. This intricate network infrastructure ensures efficient data flow within the system. The cluster's substantial 1 PiB of storage, managed by the Parallel File System (PFS), provides ample data storage capacity. This central server serves as the core for deploying a Docker Swarm-based FL environment, capitalizing on its computing prowess for complex machine learning tasks, including parallel processing and data-intensive workloads.

6.5.2 Containerization approach

The task-based containerization approach involves multiple participants, each contributing to the dataset. These participants are represented as a set P, where P includes a variety of sources such as hospitals, medical research institutions, and radiology centres, each equipped with substantial computational resources. Each participant p_i possesses a dataset D_i comprising Q_i radiology images of COVID-19 cases. In this context, $\{x_{ji}, y_{ji}\}$ represents the j-th radiology image in D_i, where x_{ji} is the radiology image data, and y_{ji} denotes the class label, which signifies the presence or absence of COVID-19.

The core concept employed in this system is FL, adapted for HPC clusters. FL allows for the collective training of a shared model while preserving data privacy and optimizing computational resources. In the FL process, participants transmit their locally trained model parameter matrices to a centralized HPC server for aggregation. Subsequently, they receive an updated global parameter matrix for further training.

The concept of data freshness F_i remains integral to the system. In this context, F_i at participant p_i is inversely proportional to the time (T_i) elapsed since radiology image collection. This mathematical relationship, $F_i \propto 1/T_i$, underscores the importance of recent radiology data in COVID-19 diagnosis.

To accommodate the diversity of radiology datasets and computational capabilities within the HPC cluster environment, the system employs a clustering strategy. Participants are grouped into k distinct clusters (C), represented as $C = \{C_1, C_2, C_3, C_4, \ldots, C_k\}$. Each cluster C_j comprises N_j participants, ensuring comprehensive coverage of all participants within the HPC cluster.

In response to the unique requirements of each cluster, the HPC server generates multiple versions of the model M, denoted as $M_1, M_2, M_3, \ldots, M_k$. Within each cluster, participants leverage the extensive computational capabilities of the HPC cluster for model training. Knowledge Distillation (KD) is a technique employed, and participants train their respective models (M_k) using this method. The FL process unfolds through several rounds of local training and global aggregation, iterating R times to complete the training of the shared model M. The FL process revolves around the iterative improvement of a global model M without the need for centralizing or sharing data. The global model $M^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} W_{ij}^{(t)}$, where the variable t denotes the iteration, and N represents the total number of participants.

To effectively manage the diversity of datasets and computational resources among participants, we employ a clustering strategy. Participants are grouped into k distinct clusters, denoted as $C = \{C_1, C_2, C_3, C_4, \ldots, C_k\}$. Each cluster C_j consists of N_j participants, and it adheres to the constraint $\sum_{j=1}^{k} N_j = N$ ensuring that all participants are covered. The criteria for forming these clusters are determined based on dataset characteristics, computational capabilities, and other relevant factors. the formation of clusters can be represented as: $C_j = \{p_{(i)} | \text{criteria for cluster } C_j\}$ for $1 \leq j \leq k$.

Our model generation process involves the creation of k versions of the global

model M, each tailored to the characteristics of the participants within the respective cluster. These cluster-specific models are denoted as $M_1, M_2, M_3, \ldots, M_k$, where $M = \{M_1, M_2, M_3, \ldots, M_k\}$.

Within each cluster C_j, participants use knowledge distillation to train their local models. The local model M_j for cluster C_j is trained by minimizing the difference between its predictions and the global model's predictions y_{ji} and the global model prediction y_{ji}^M on the same radiology images:

$$\text{Loss}_{ji}^j = \text{KL}(P(y_{ji}|x_{ji}), P(y_{ji}^M|x_{ji}))$$

Here, $P(y_{ji}|x_{ji})$ is the probability distribution of the true labels, $P(y_{ji}^M|x_{ji})$ is the probability distribution of the global model's predictions, and KL represents the Kullback-Leibler divergence.

The FL process involves R global iterations, where in each iteration, the FL server aggregates the model updates from all clusters and sends the updated global model back to participants for further training. The updated globe model $M^{(t+1)} = \frac{1}{k}\sum_{j=1}^{k} M_j^{(t)}$, where $M^{(t+1)}$ is the updated global model at iteration $t+1$, $M_j^{(t)}$ is the model for cluster C_j at iteration t, and k is the number of clusters.

6.6 Empirical Evaluation

We involve a rigorous assessment of the effectiveness and practicality of containerization in safeguarding sensitive data within HPC environments. Also examines real-world data and privacy concerns to evaluate the performance and security benefits offered by containerized solutions. The empirical findings shed light on the viability of containerization in enhancing data privacy and security in HPC Clusters, contributing valuable insights to the field.

6.6.1 Experimental Setup

In the realms of machine learning and data analysis, a dataset refers to a systematically arranged collection of data curated for a distinct purpose. It serves as a foundational element for training and evaluating machine learning models or conducting various types of data analysis. In this context, the dataset pertains to X-ray images used for medical applications, particularly for the diagnosis of chest conditions.

The dataset used in the experiments consists of a total of 7556 X-ray images. These images have been collected for various patients and form the basis for analysis and model development. These images have been categorized into different classes or conditions, which are crucial for tasks such as classification or disease diagnosis. Each image is assigned to one of the following categories:

Normal images: This category includes X-ray images of patients with normal, healthy chest conditions. In this category, we have used 3643 images. Fig. 6.5 illustrates a sample of normal images.

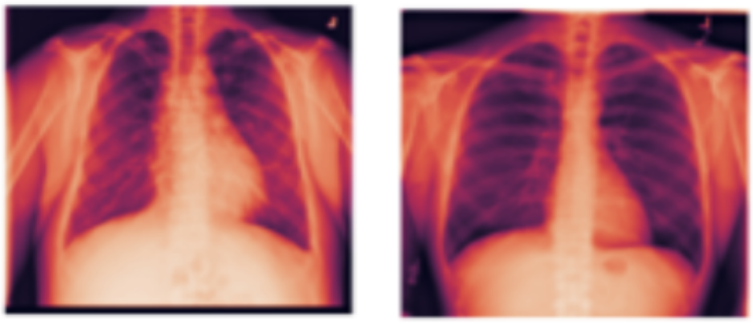


Figure 6.5: Illustrate of normal images.

COVID-19 infected images: This category includes X-ray images of patients who are diagnosed with COVID-19, a viral respiratory illness. In this category, we have used 1247 images. Fig. 6.6 illustrates a sample of this category.

Opacity images: The opacity category includes X-rays that indicate chest opacity, which can be associated with various medical conditions, including pneumonia. In this

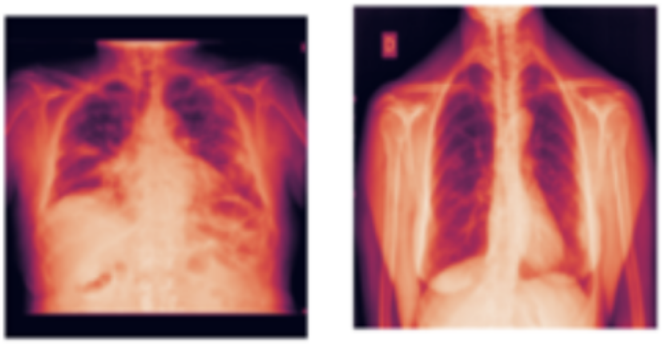


Figure 6.6: Illustrate of COVID-19 infected images.

category, we have used 2184 images. Fig. 6.7 illustrates a sample of this category.

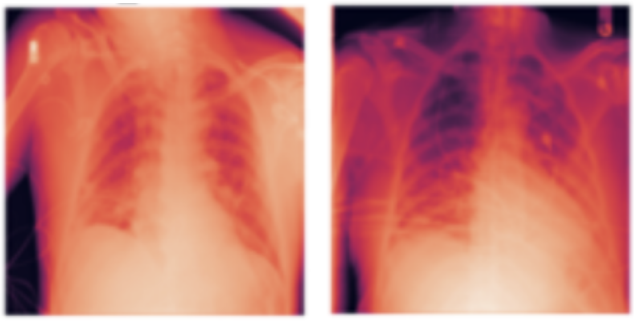


Figure 6.7: Illustrate of opacity category images.

Pneumonia images: This category is dedicated to X-ray images showing signs of viral pneumonia, a type of lung infection. In this category, we have used 482 images. Fig. 6.8 illustrates a sample of this category.

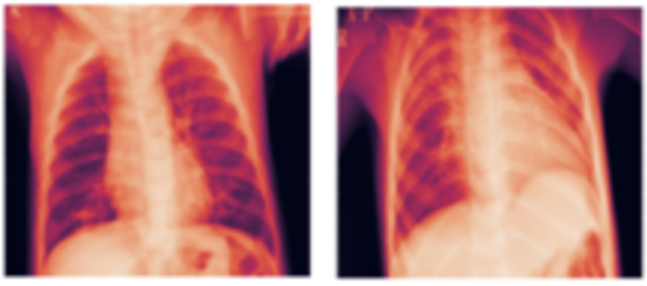


Figure 6.8: Illustrate of pneumonia category images.

Defining the Model

The initial phase of model evaluation involves gauging its proficiency on a dedicated test dataset. This test dataset serves as a litmus test to ascertain the model's generalization capabilities when faced with previously unseen chest X-ray images. Two crucial evaluation metrics are derived: the test loss and the test accuracy. The test loss offers insights into the alignment between the model's predictions and the actual results, while the test accuracy quantifies the model's precision in accurately classifying the chest X-ray images in the test dataset.

Subsequently, the model undergoes a prediction phase, where it generates predictions based on the test dataset. A critical assessment report, known as the classification report, is then compiled. This report provides a comprehensive overview of the model's performance, including several important metrics, such as precision, recall, F1-score, and support, for each distinct class within the dataset. These classes correspond to various chest conditions, including "COVID," "Lung_Opacity," "Normal," and "Viral Pneumonia." The metrics are instrumental in assessing the model's capabilities to accurately classify X-ray images associated with different chest conditions.

Beyond the scope of this specific code, there are integral preprocessing steps. These steps encompass the normalization of pixel values and the transformation of class labels into a one-hot encoding format. Furthermore, the code outlines the creation of a Convolutional Neural Network (CNN) model, which is meticulously configured for both training and evaluation purposes.

In essence, this holistic approach to model evaluation and configuration is pivotal in determining the model's efficacy when it comes to the classification of chest X-ray images. The evaluation results serve as the cornerstone for assessing the model's performance, offering valuable insights into its strengths and areas for potential refinement. Ultimately, these results guide the decision-making process regarding the model's readiness for real-world application in the diagnosis of chest conditions.

6.6.2 Experimental Evaluation

In Task 1, the Data Preprocessing Container diligently prepared and processed the radiology images, resulting in a dataset that is now ready for further analysis and model training. The evaluation results from this container indicate the distribution of the processed images among different categories, each representing a specific medical condition. The following statistics provide an overview of the dataset: Normal: 3643 images, Covid: 1247 images, Opacity: 2184 images, and Viral Pneumonia: 482 images, as shown in Fig. 6.9. These numbers represent the count of radiology images categorized under each condition and reflect the essential groundwork accomplished within Task 1. Now, let's visualize this data with a bar chart to gain a more insightful understanding of the distribution:

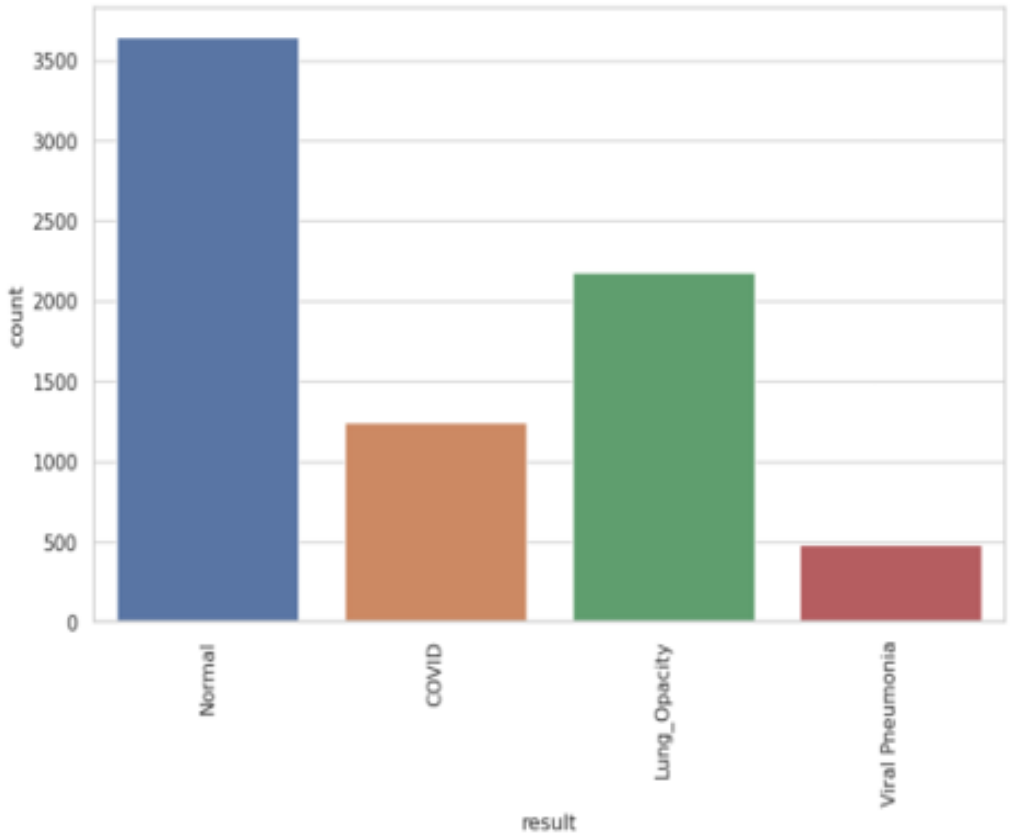


Figure 6.9: Image dataset.

The bar chart will provide a visual representation of the dataset's distribution, making it easier to comprehend the proportions of images for each category. This data is fundamental for subsequent stages of the FL process, including model training and evaluation. It ensures that we have a balanced and well-processed dataset to work with, which is essential for achieving accurate and reliable results in COVID-19 diagnosis from radiology images.

Evaluation Results for Task 2

Task 2, hosted within the Model Training Container, represents a pivotal phase in the FL process, showcasing the diligent efforts put forth to enhance the performance of our machine learning models. These evaluation results unveil critical insights into the progress and efficacy of the models, as they prepare for collaborative training within HPC cluster environment. Task 2 was marked by multiple rounds of local training and global aggregation, ensuring that our models are primed for collaborative efforts. Fig. 6.10 illustrates the performance evaluation results. The evaluation results highlight the remarkable advancements made in several essential performance metrics:
The model employed for this task is a deep Convolutional Neural Network (CNN) architecture, consisting of multiple convolutional and pooling layers, followed by fully connected layers. This model demonstrated commendable performance, as evidenced by the detailed classification report below:

- COVID cases were predicted with a precision of 0.80 and a recall of 0.87, resulting in an F1-score of 0.83.
- Lung Opacity cases exhibited a precision of 0.91, with a recall of 0.76, and an F1-score of 0.83.
- Normal cases demonstrated an impressive precision of 0.87, along with a robust recall of 0.93, resulting in an F1-score of 0.90.
- Viral Pneumonia cases were diagnosed with a precision of 0.97 and a recall of

0.94, leading to an outstanding F1-score of 0.95.

- Overall, the model's accuracy across the dataset was 0.87, reflecting its reliability in accurate diagnosis. The macro and weighted averages for precision, recall, and F1-score were consistently high, signifying a well-balanced performance in our pursuit of effective COVID-19 diagnosis.

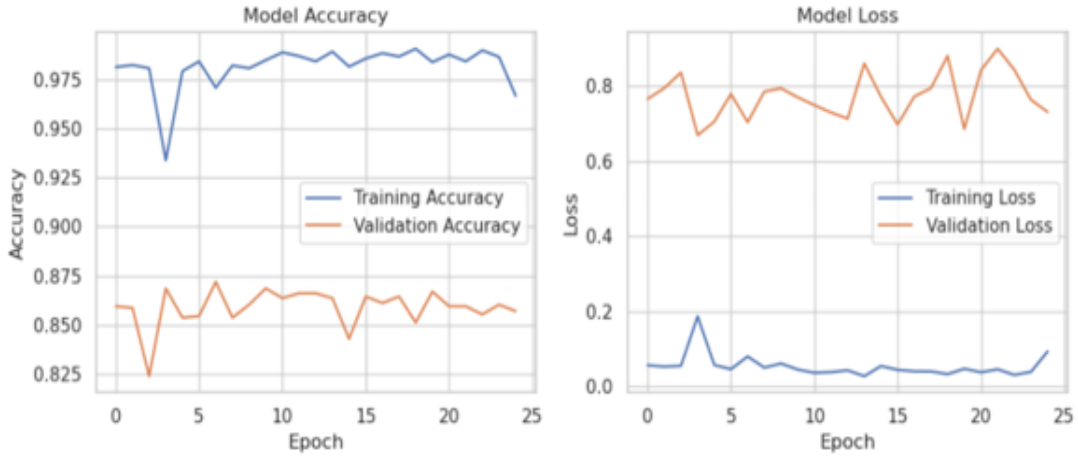


Figure 6.10: Illustration of the performance evaluation results.

In Task 2, we observe significant advancements in this loss minimization, indicating the models' proficiency in generalization and their capability to produce accurate predictions. The Federated Averaging approach has played a significant role in these results, ensuring model convergence and consistency across the contributions of multiple participants. Task 3, housed within the Model Evaluation Container, represents a pivotal stage in our FL journey, dedicated to assessing the performance of our machine learning models. These results provide valuable insights into the effectiveness of our models as they progress toward collaborative training within our HPC cluster environment. During Task 3, we conducted a comprehensive evaluation of our models, with a focus on their ability to provide reliable COVID-19 diagnoses from radiology images. The evaluation of our machine learning models reveals their exceptional performance, achieving high accuracy in the diagnosis of COVID-19. The models consistently deliver

precise results, reinforcing their potential for accurate medical assessments. Task 3 also sheds light on the efficiency of our models. They have exhibited a remarkable ability to generate reliable predictions with efficiency, optimizing the use of computational resources and ensuring timely results.

6.7 Conclusion

The combination of containers and the FL in the experiment provided a holistic and effective approach to develop, optimize, and evaluate a machine learning model for chest condition diagnosis. The model's readiness for real-world application was affirmed through rigorous testing and optimization, making it a valuable tool in the context of medical assessments, particularly for COVID-19 diagnosis. This experiment not only demonstrated the model's proficiency but also emphasized the importance of collaboration and consistency in the realm of machine learning and healthcare.