Literature Review

Architectural Patterns for the Design of Federated Learning Systems

Overview of architectural patterns for federated learning

Category	Name	Summary
Client management patterns	Client registry	Maintains the information of all the participating client devices for client management.
	Client selector	Actively selects the client devices for a certain round of training according to the predefined criteria to increase model performance and system efficiency.
	Client cluster	Groups the client devices (i.e., model trainers) based on their similarity of certain characteristics (e.g., available resources, data distribution, features, geolocation) to increase the model performance and training efficiency.
Model management patterns	Message compressor	Compresses and reduces the message data size before every round of model exchange to increase the communication efficiency.
	Model co-versioning registry	Stores and aligns the local models from each client with the corresponding global model versions for model provenance and model performance tracking.
	Model replacement trigger	Triggers model replacement when the degradation in model performance is detected.
	Deployment selector	Selects and matches the converged global models to suitable client devices to maximise the global models' performance for different applications and tasks.
Model training patterns	Multi-task model trainer	Utilises data from separate but related models on local client devices to improve learning efficiency and model performance.
	Heterogeneous data handler	Solves the non-IID and skewed data distribution issues through data volume and data class addition while maintaining the local data privacy.
	Incentive registry	Measures and records the performance and contributions of each client and provides incentives to motivate clients' participation.
Model aggregation patterns	Asynchronous aggregator	Performs aggregation asynchronously whenever a model update arrives without waiting for all the model updates every round to reduce aggregation latency.
	Decentralised aggregator	Removes the central server from the system and decentralizes its role to prevent single-point-of-failure and increase system reliability.
	Hierarchical aggregator	Adds an edge layer to perform partial aggregation of local models from closely-related client devices to improve model quality and system efficiency.
	Secure aggregator	The adoption of secure multiparty computation (MPC) protocols that manages the model exchange and aggregation security to protect model security.

1. Client Manager Patterns

Client management patterns describe the patterns that manage the client devices' information and their interaction with the central server.

1.1 Client registry

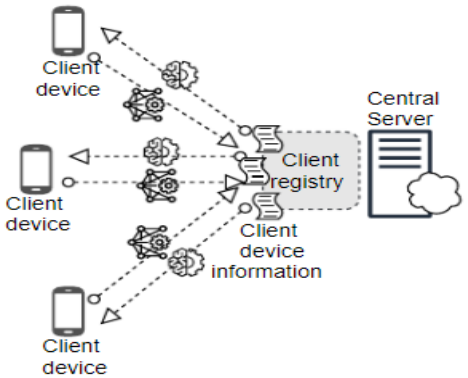

Figure 4: Client Registry.

The central server sends the request for information to the client devices. The client devices then send the requested information together with the first local model updates.

1.2 Client selector

A client selector actively selects the client devices for a certain round of training according to the pre-defined criteria to increase model performance and system efficiency.

Context:

Multiple rounds of model exchanges are performed and communication cost becomes a bottleneck. Furthermore,multiple iterations of aggregations are performed and consume high computation resources.

Problem:

The central server is burdensome to accommodate the communication with massive number of widely-distributed client devices every round.

Forces:

The problem requires the following forces to be balanced:

- Latency. Client devices have system heterogeneity (difference in computation, communication, & energy resources)that affect the local model training and global model ag-gregation time.
- Model quality. Local data are statistically heterogeneous(different data distribution/quality) which produce local models that overfit the local data.

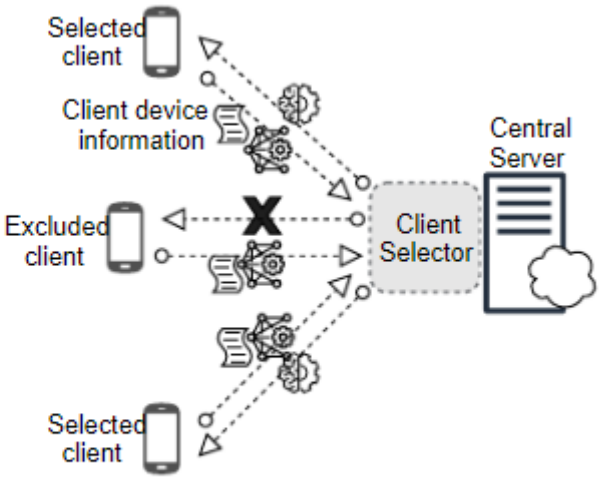

Figure 5: Client Selector.

Solution:

Selecting client devices with predefined criteria can optimize the formation of the global model. The client selector on the central server performs client selection ev-ery round to include the best fitting client devices for global model aggregation. The selection criteria can be configuredas follows:

- Resource-based: The central server assesses the resources available on each client devices every training round and selects the client devices with the satisfied resource status(e.g., WiFi connection, pending status, sleep time)
- Data-based: The central server examines the information of the data collected by each client, specifically on the number of data classes, distribution of data sample vol-ume per class, and data quality. Based on these assess-ments, the model training process includes devices with high-quality data, higher data volume per class, and excludes

the devices with low-quality data, or data that are highly heterogeneous in comparison with other devices.

- Performance-based: Performance-based client selection can be conducted through local model performance as-sessment (e.g., performance of the latest local model or the historical records of local model performance)

Consequences:

Benefits:

- Resource optimisation. The client selection optimizes the resource usage of the central server to compute and com-municate with suitable client devices for each aggregation round, without wasting resources to aggregate the low-quality models.
- System performance. Selecting clients with sufficient power and network bandwidth greatly reduces the chances of clients dropping out and lowers the communication latency.
- Model performance. Selecting clients with the higher lo-cal model performance or lower data heterogeneity in-creases the global model quality.

Drawbacks:

- Model generality. The exclusion of models from certain client devices may lead to the missing of essential data features and the loss of the global model generality.
- Data privacy. The central server needs to acquire the sys-tem and resource information (processor's capacity, net-work availability, bandwidth, online status, etc.) every round to perform devices ranking and selection. Access to client devices' information creates data privacy issues.
- Computation cost. Extra resources are spent on transfer-ring of the required information for selection decision-making

1.3 Client Cluster

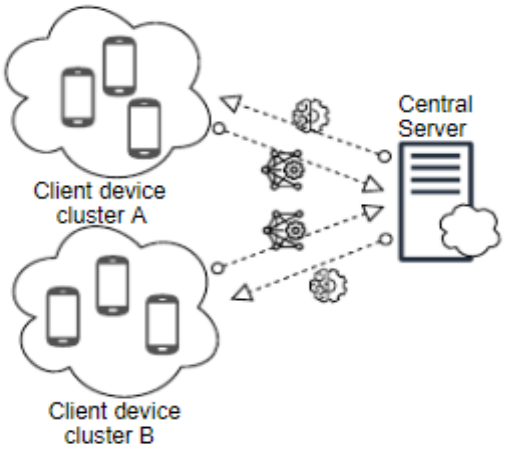

Figure 6: Client Cluster.

2. Model Management Patterns

Model management patterns include patterns that handle model transmission, deployment, and governance.

2.1 Message Compressor

A message compressor reduces the transmitted message size.

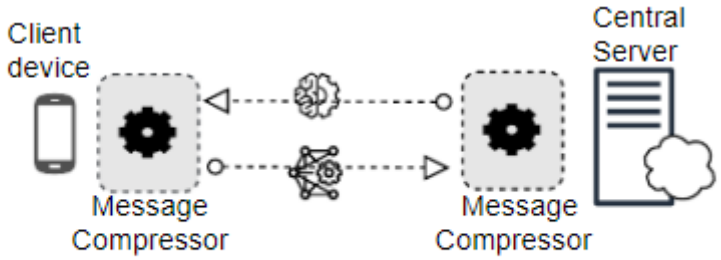

Figure 7: Message Compressor.

2.2 Model Co-versioning Registry

A model co-versioning registry records all local model versions from each client and aligns them with their corresponding global model.

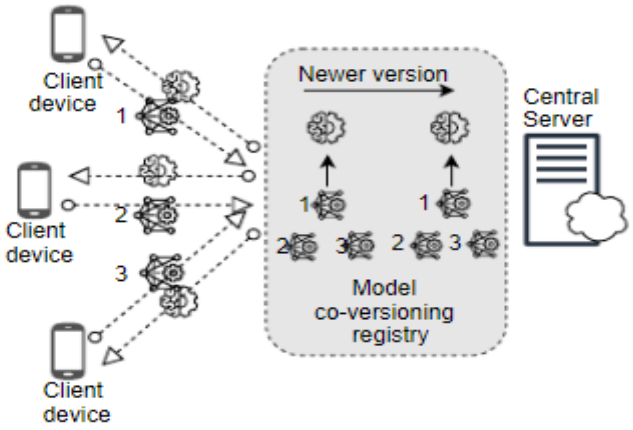

Figure 8: Model Co-versioning Registry.

2.3 Model Replacement Trigger

A model replacement trigger initiates a new model training task when the converged global model's performance degrades.

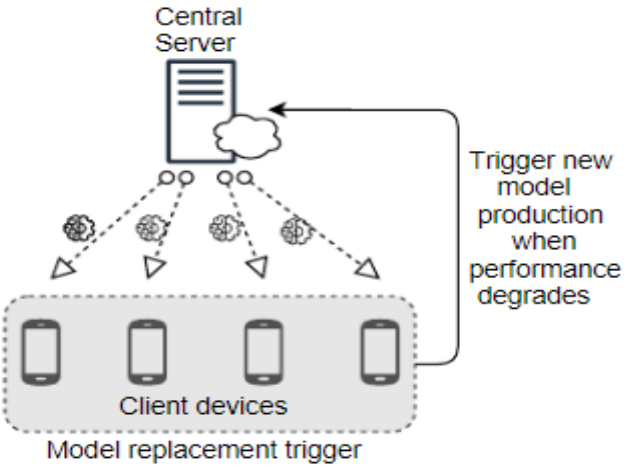

Figure 9: Model Replacement Trigger.

Fig. 9illustrates a model replacement trigger that initiates a new model training task when the current global model's performance drops below the threshold value or when a degrade on model prediction accuracy is detected.

Context:
The client devices use converged global model for inference or prediction

Problem:
As new data is introduced to the system, the global model accuracy might reduce gradually. Eventually, with the degrading performance, the model is no longer be suitable for the application.

Forces:
The problem requires to balance the following forces:•Model quality. The global model deployed might expe-rience a performance drop when new data are used for inference and prediction.

- Performance degradation detection. The system needs to effectively determine the reason for the global model's performance degradation before deciding whether to acti-vate a new global model generation

Solution:
A model replacement trigger initiates a new model training task when the acting global model's performance drops below the threshold value. It will compare the per-formance of the deployed global model on a certain number of client devices to determine if the degradation is a global event. When the global model performance is lower than the preset threshold value for more than a fixed number of consecutive times, given that performance degradation is a global event, a new model training task is triggered.

Consequences:

Benefits:

- Updatability. The consistent updatability of the global model helps to maintain system performance and reduces the non-IID data effect. It is especially effective for clients that generate highly personalized data that causes the ef-fectiveness of the global model to reduce much faster as new data is generated.

- Model quality. The ongoing model performance monitor-ing is effective to maintain the high quality of the global model used by the clients.

Drawbacks:

- Computation cost. The client devices will need to perform model evaluation periodically that imposes extra computational costs.
- Communication cost. The sharing of the evaluation results among clients to know if performance degradation is a global event is communication costly.

2.4 Deployment Selector

A deployment selector deploys the global model to the selected clients to improve the model quality for personalized tasks.

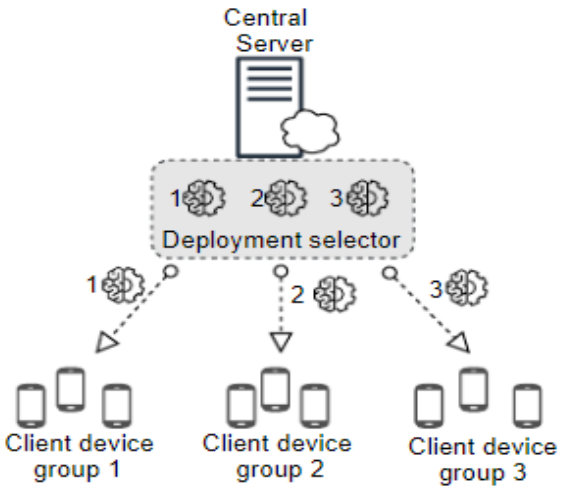

Figure 10: Deployment Selector.

3. Model Training Patterns

3.1 Multi-task Model Trainer

3.2 Heterogeneous Data Handler

Solution:

A heterogeneous data handler balances the data distribution and solves the data heterogeneity issue in the client devices through data augmentation and federated distillation. Data augmentation solves data heterogeneity by generating augmented data locally until the data volume is the same across all client devices. Furthermore, the classes in the datasets are also populated equally across all client devices. Federated distillation enables the client devices to obtain knowledge from other devices periodically without directly accessing the data of other client devices. Other methods includes taking the quantified data heterogeneity weightage (e.g, Pearson's correlation, centroid averaging-distance etc.) into account for model aggregation.

4. Model Aggregation Patterns

Model aggregation patterns are design solutions of model aggregation used for different purposes. Asynchronous aggregator aims to reduce aggregation latency and increase system efficiency, whereas decentralized aggregator targets to increase system reliability and accountability. Hierarchi-cal aggregator is adopted to improve model quality and op-timises resources. Secure aggregator is designed to protect the models' security.

4.1 Asynchronous Aggregator

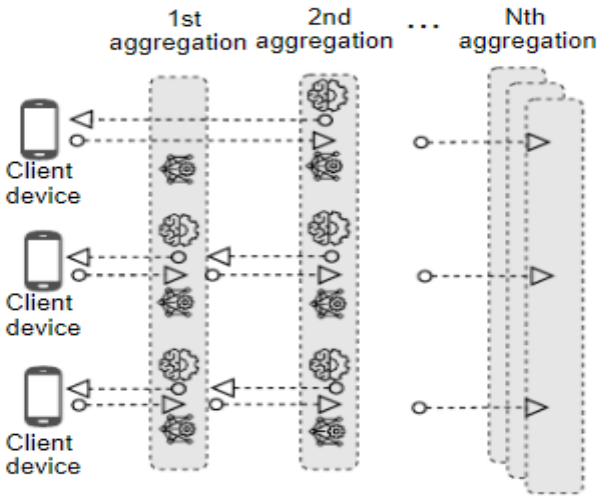

Figure 14: Asynchronous Aggregator.

Problem:

Due to the difference in computation resources,the model training lead time is different per device. Further-more, the difference in bandwidth availability, communica-tion efficiency affects the model's transfer rate. Therefore,the delay in model training and transfer increases the latency in global model aggregation.

4.2 Decentralized Aggregator

A decentralized aggregator replaces the central server's role in a federated learning system. The aggregation and update of the models can be performed through peer-to-peer exchanges between client devices. First, a random client from the system can be an aggregator by requesting the model updates from the other clients that are close to it.Simultaneously, the client devices conduct local model train-ing in parallel and send the trained local models to the aggre-gator. The aggregator then produces a new global model and sends it to the client network. Blockchain is the alternativeto the central server for model storage that prevents single point-of-failure. The ownership of the blockchain belongs to the learning coordinator that creates the new training tasks and maintains the blockchain. Furthermore, the record of models on a blockchain is immutable that increases the reli-ability of the system. It also increases the trust of the system as the record is transparent and accessible by all the client devices.

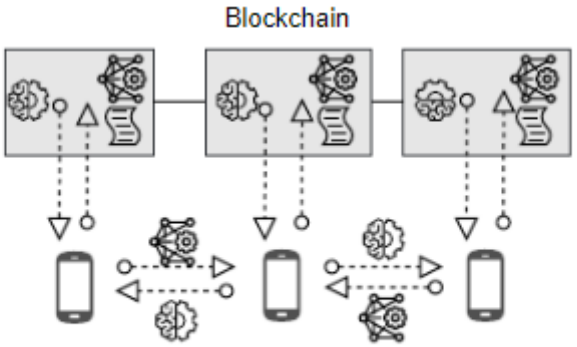

Figure 15: Decentralised Aggregator.

4.3 Hierarchical Aggregator

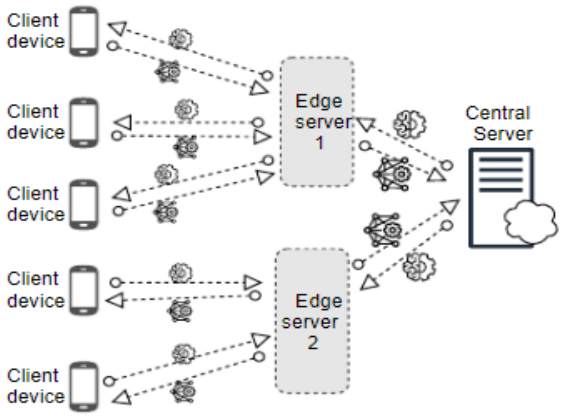

Figure 16: Hierarchical Aggregator.

To reduce non-IID effects on the global modeland increase system efficiency, a hierarchical aggregator adds an intermediate layer (e.g., edge server) to perform partial aggregations using the local model parameters from closely-related client devices before the global aggregation.

In Fig. 16,edge servers are added as an intermediate layer between the central server and client devices to serve the client devices that are closer to them.

4.4 Secure Aggregator

A security aggregator handles the secure mul-tiparty computation (SMC) protocols for model exchangesand aggregations. The protocols provide security proof to guarantee that each party knows only its input and output.For instance, homomorphic encryption is a method to en-crypt the models and only allow authorized client devices and the central server to decrypt and access the models. Pair-wise masking and differential privacy (DP) methods are ap-plied to reduce the interpretability of the model by unautho-rised party [46]. The technique involves adding noise to the parameters or gradient or uses a generalized method.

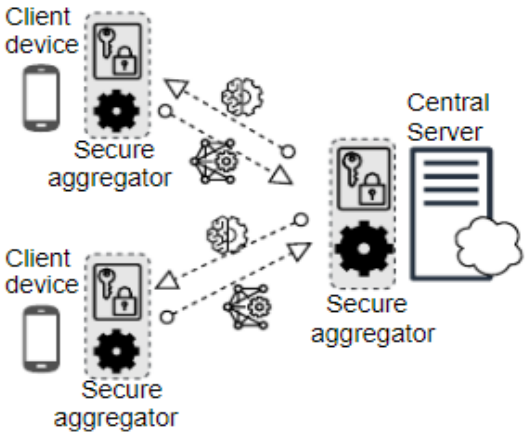

Figure 17: Secure Aggregator.