ONNX
Distributed learning

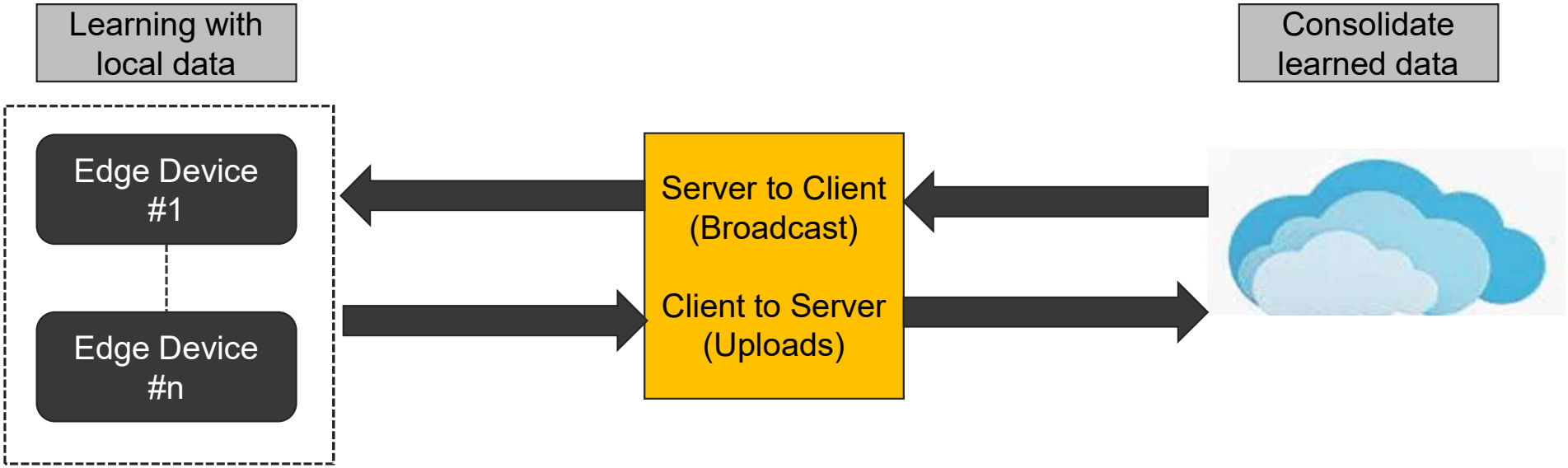
Presenter(s): Rodolfo Esteves, Rajeev Nalawadi
Intel Corporation

Distributed Learning Trends

Distributed Learning on edge /client using local data and later consolidation of learned parameters in the cloud is an increasing trend due to some of the driving factors:

- Privacy of data
- Data Sovereignty
- Bandwidth concerns
- Latency
- Security
- Costs of data centralization
- Model compression [*]
- Access to decentralized data [*]

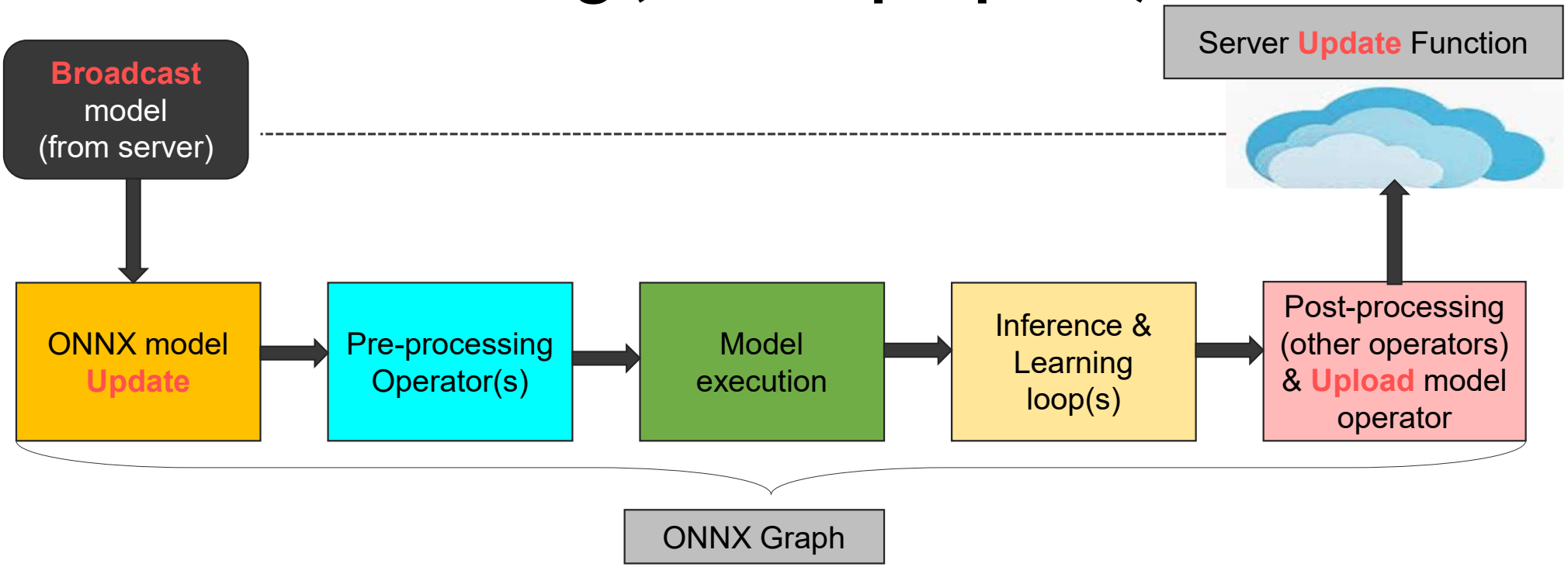
Distributed learning flows (Edge <-> Cloud)



Distributed Learning Elements with communication collectives

- Server-to-Client broadcast step
- Local client update step
- Client-to-Server upload step
- Server update step

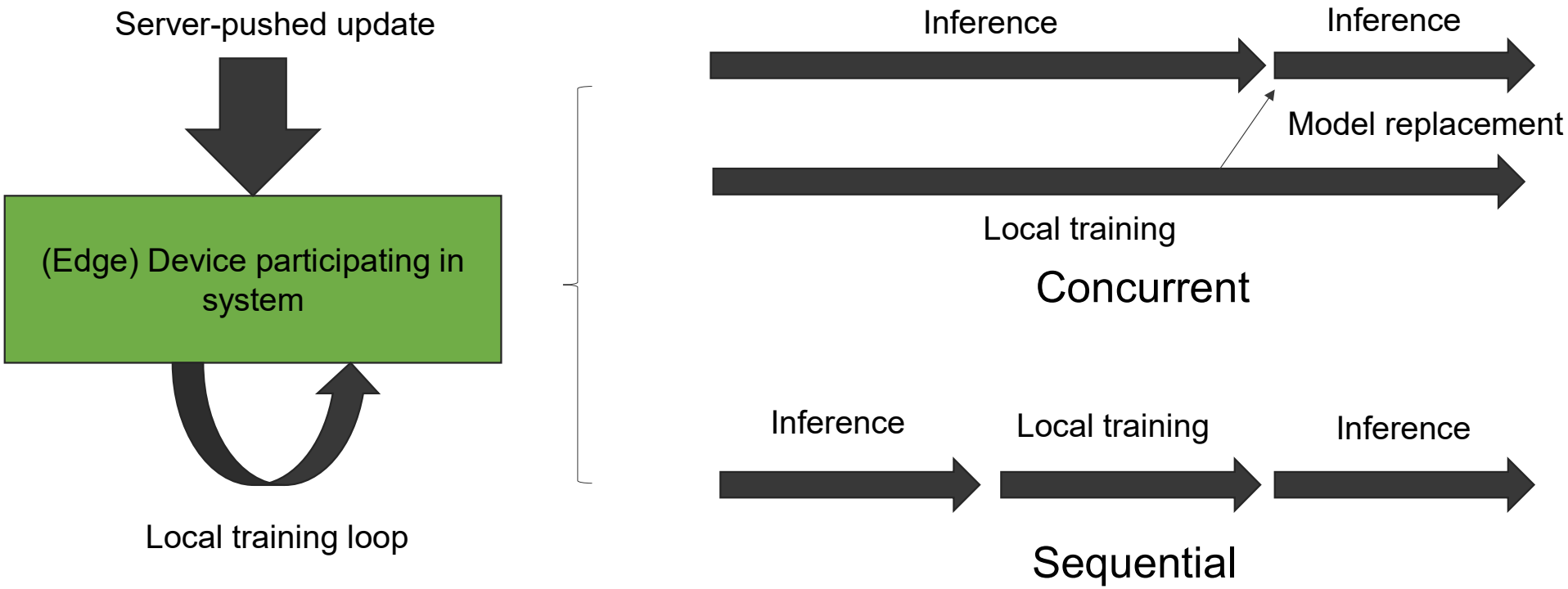
Distributed learning (ONNX proposal)



ONNX scope (suggested)

- Consider communications as the responsibility of the application
- Concentrate on local loop

Distributed learning: client loop(s)



New ONNX operators as Functions (requested)

- Query parameters of local model
- Local client update function
- Metadata to indicate what layers can be updated

Federated learning (some thoughts)

Some initial thoughts for consideration :

- For heterogeneous clients, what optimizations on the model (if any) are allowed (eg: local model quantizations) ?
- Many updates / clients boarding or dropping / tolerance to latency / resilience (e.g. for data poisoning) / other decision points are possible and can be application-specific, support should be flexible to accommodate these
- How much of this can be built on top of current ONNX using functions (as a library or application architecture) and how much needs to change the spec
- Suggested design: allow for local nodes to run different runtimes and possibly different serializations of the same architecture (as long as parameters are the same)
- Can probably schedule a plan to incrementally gain capabilities, that combined together allow federated learning

Thank You !!