Project: NDN Compute Simulator (ndnCSIM)

11th NDN Hackathon

Project Lead: Muhammad Atif Ur Rehman

Team Members: Muhammad Salah ud Din, Sana Fayyaz, Muhammad Imran

Broadband Convergence Networks Laboratory,

Hongik University, Republic of Korea.

What have we solved and Why?

- In-network computations and Edge Cloud Computing
- NDN community is actively exploring the in-network computation.
- **ndnSIM**: the most popular simulator for NDN
- Unfortunately, ndnSIM does not support in-network computation simulations

ndnCSIM, a kind of compute toolkit, provide the basic implementation for service-based computations in ndnSIM

Tasks Achieved This Project

- NDN Apps
  - NDN Consumer Compute Applications
  - NDN Producer Compute Application
- Modifications in NFD
  - forwarder.cpp and forwarder.hpp files now contains the necessary functions to perform computations
- The node.h file in Network/model is modified
  - m_compute_capacity
  - m_initial_compute_capacity
  - m_inital_compute_resources
  - m_compute_resources

NDN APPS

- *ConsumerCompute* class
  - Updated the name by adding service information. For instance, it generate the name as follows: **"/Prefix/ | service_1/1/1"**
- *ProducerCompute* class
  - Functions related to **Service Information Collection**

```
int
AddToServiceSatusCollection(string service_name, int start_time, int node_id,int run_time,ServiceStatus service_status);

virtual ProducerCompute::ServiceInfo
UpdateServiceStatus(int service_id, ProducerCompute::ServiceStatus updated_service_status);

virtual ProducerCompute::ServiceInfo
FetchFromServiceCollectionByNameId(string service_name, int service_id);

virtual ProducerCompute::ServiceInfo
FetchFromServiceCollectionById(int service_id);

virtual vector<ProducerCompute::ServiceInfo>
FetchFromServiceCollectionByName(string service_name);

virtual vector<ProducerCompute::ServiceInfo>
FetchFromServiceCollectionByStatus(ProducerCompute::ServiceStatus service_status);

virtual int
GetServiceIndexInCollection(int service_id);

virtual tuple<string,string,string>
GetServiceInformation(shared_ptr<const Interest> interest);
```

NDN APPS

- *ProducerCompute* class
  - Functions related to **Service Execution**

```
enum ServiceStatus {
        Running = 1,
        Paused = 2,
        Completed = 3
};

struct ServiceInfo
{
  int service_id; //
  string service_name;
  int start_time; // in milisecond
  int node_id;
  int run_time;
  ProducerCompute::ServiceStatus service_status;
  double resource_utilization_value;
};
vector<ProducerCompute::ServiceInfo> service_info_collection;

tuple<int,int,int,int>
StartServiceExecution(shared_ptr<const Interest> interest);

virtual void
EndServiceExecution(int service_id,int resource_value);

virtual void
PauseServiceExecution(int service_id);

virtual int
GetServiceExecutionTime(int service_type);
```

NDN APPS

- *ProducerCompute* class
  - Functions related to **Node's resource utilizations**

```
virtual int
OccupyNodeResources(int service_type);

virtual void
ReleaseNodeResources(int resource_value);

virtual double
GetComputeResourceUtilizationValue(int service_type);
```

FORWARDER CLASS MODIFICATIONS

- Forwarder is also modified
  - Provide all the functions similar to ProducerCompute class
  - The computations are started when the cache hit occur
    - However, it depends on the research problem and one can change the execution logic as per requirement
  - Enable the intermediate nodes to simulate the computation behavior

ADVANTAGES OF NDN COMPUTE SIMULATOR TOOLKIT

- Provide basic compute functions for simulations
- Monitor the nodes resources utilizations
- Offloading compute tasks based on the resource availability
  - By checking the available resources of the nodes
- A complete history of services executed on each node during the simulation time

DEMO