Fog Computing: Concepts, Challenges and Research Scope

Satish Srirama
satish.srirama@ut.ee
Outline

• Fog computing
• Fog computing challenges
• Fog placement strategies
• Future research directions
Layers of Cloud-centric IoT

Remote Cloud-based processing

Connectivity nodes & Embedded processing

Sensing and smart devices
Issues with Cloud-centric IoT

• The cost of a cloud-only solution is too high
  – To run a large scale IoT system with >1000 geo-distributed devices [FogFlow]
• Latency issues for applications with sub-second response requirements
  – Health care scenarios
  – Smart cities and tasks such as surveillance need real-time analysis with strict deadlines
  – Many IoT services require <10ms end-to-end latency
• Network load
• Certain scenarios do not let the data move to cloud
  – Better security and deeper insights with privacy control
Fog Computing

• Processing across all the layers, including network switches/routers

[Chang et al, AINA 2017; FEC 2019; Mass et al, SCC 2016; Liyanage et al, PDCAT 2016]
Fog applications

- Multi-media applications
  - Finding best data streaming bit rate in different scenarios
    - Such as in video surveillance applications
  - Adjusting video encoding rate (video processing speed) based on network load
- Sensor data filtering and preprocessing
- Sensing frequency calibration
- Sensor data prediction
- Interactive games
  - Electroencephalogram (EEG) Tractor Beam game
  - Real-time EEG signal analysis and brain state (concentration) prediction

[Zao et al, Frontiers in Human Neuroscience 8:370]
Fog Architecture

- **Large-scale distributed data analytics**
  - e.g. MapReduce, Flink, Spark
- **Edge Analytics**
  - Filtering, consolidation, error detection, anonymization
- **Cloud**
- **Core Network**
  - Fog Nodes
    - Private Clouds, Cisco IOx, Switches, Cloudlets
  - Gateways
    - Access points, Mobile Phones, Single Board Computers, CPE
  - Edge Nodes
    - Sensors, Actuators, Mobile Phones, Smart Vehicles, ...

11/12/2019
Satish Srirama
Advantages of Fog Computing

• **Security**
  – Supports additional security to IoT devices to ensure safety and trustworthiness in transactions

• **Cognition**
  – Enables fog providers the awareness of the objectives of their clients toward supporting autonomous decision-making
    • E.g. where and when to deploy computing, storage, and control functions

• **Agility**
  – Brings the opportunity to individual and small businesses to participate in providing FEC services

• **Latency**
  – Provide rapid responses for the applications that require ultra-low latency

• **Efficiency**
  – Reduces the unnecessary cost of outgoing communication bandwidth
  – Consume minimum power for data offloading and processing when compared to CIoT model

[Chang et al, FEC 2019]
Types of fog computing applications

• Top-down applications
  – Applications that are managed by a cloud provider
  – Cloud provider deploys the necessary geo-distributed resources
  – Cloud providers manage the IoT application execution across the fog topology
  – Consider proximity and QoS parameters such as user load, latency, cost models etc.

• Bottom up applications
  – Applications that are managed by individual service providers
  – Fog resources are provided by different vendors
    • E.g. General public, private clouds, network operators
  – Gateways decide how to schedule the components across the fog topology
Fog Service Types

• **Storage (caching)**
  – Keep tenant-side client data locally
  – Temporary caching periodical data
  – E.g. Static local healthcare and eldercare

• **Compute (VM/ containers; IaaS/PaaS/FaaS / SaaS/ CaaS/ )**
  – Pre-processing data before sending to the cloud
  – Predefined rules deployed at the fog nodes

• **Acceleration (GPU/FPGA/APU etc.)**
  – Fog nodes with Tensor Processing Unit (TPU), AI Processing Unit (APU) etc.

• **Networking**
  – Support for Software-defined networking (SDN) / Network functions virtualization (NFV)

• **Control**
  – Deploy, actuating, mediating, security etc.
Research Challenges in Fog Computing

- Frameworks for establishing fog setups
- Fog resource provisioning
- Dynamic fog computing service discovery and accessing
- Fog execution frameworks
- Fog application placement
- Mobility of fog devices
- Etc.
IndieFog

- Indie Fog [Chang et al, IEEE Computer 2017]
  - System architecture for enabling fog computing with customer premise equipment
- Proactive fog computing using resource-aware work-stealing [Soo et al, IJMCMC 2017]
- Dynamic fog computing service discovery and accessing
Fog execution frameworks – Actor programming model

- Distributed and fault-tolerant execution of Fog computing applications
- Based on Actor programming model
- Actor model is conceived as a universal paradigm for concurrent computation [Carl Hewitt, 1973]
  - Basic unit of computation is an Actor
  - Actors communicate with other actors through messages
  - Actor can also create other actors (Parent-child hierarchy)
  - Designate the behavior to be used for the next message it receives
- Have implemented applications using the Akka framework [Srirama et al, FGCS 2020 (Under review)]
Serverless computing

• IoT workloads are a better fit for event driven programming
  – Execute app logic in response to sensor data
  – Execute app logic in response to scheduled tasks etc.
• Event-action platforms to execute code in response to events
• Applications are charged by compute time (millisecond) rather than by reserved resources
• Serverless computing is ideal solution for fog processing
  – OpenFaaS, light-weight enough to place on Raspberry Pi
Fog application placement strategies

• Resource intensive tasks of IoT applications can be placed across the Fog topology
• How to decide to which node the task to be offloaded?
  – Based on quality of service (QoS) parameters such as latency, resources, cost etc.
  – Latency and cost-aware [Mahmud et al, JPDC 2020] application module management
• The problem can also be formulated as multi-objective offloading strategy [Adhikari et al, IEEE IoTJ 2020]
  – Latency and resource management
  – Need to consider the types of the jobs and their priority
  – Need to find ideal heuristics, metaheuristics etc.
  – Also have to consider the graph topology of the Fog nodes

QoS – Quality of Service
QoE – Quality of Experience
Application Offloading Strategy for Hierarchical Fog Environment through Swarm Optimization

- Multi-tier Fog-Cloud Model
- IoT layer
- Low-capacity fog layer
  - Resource constrained low capacity fog devices
  - Mainly process delay-sensitive applications
- High-capacity fog layer
  - Resource constrained high capacity fog devices
  - Process larger delay-sensitive applications or smaller computation-intensive applications
- Cloud layer
  - Can process any kind of computation-intensive applications

[Adhikari et al, IEEE IoTJ 2020]
QoS parameters – Cost and resource utilization

- Calculate cost of CPU and memory usage:
  
  \[ CO_{kl}^C = \frac{P_{kl} \times CO_{kl}^C}{\tau_1}, \forall (k,l) \]
  
  \[ CO_{kl}^{mm} = \frac{P_{kl} \times CO_{kl}^{mm}}{\tau_2}, \forall (k,l) \]
  
  \[ CO_{kl} = CO_{kl}^C + CO_{kl}^{mm}, \forall (k,l) \]

- Calculate communication cost:

  \[ CO_{kl}^{LD} = \frac{LD_{kl} \times CO_{kl}^{LD}}{\tau_3}, \forall (k,l) \]

- Calculate total cost:

  \[ TCO_{kl} = CO_{kl} + CO_{kl}^{LD}, \forall (k,l) \]

- Calculate resource utilization:

  \[ RU_{kl} = \frac{L_k(t)}{W_l(t)} \times 100, \forall (k,l) \]

- Formulate multi-objective function:

  \[ MO_{kl} = \Delta_1 RU_{kl} + \Delta_2 TCO_{kl}, \forall (k,l) \]

```plaintext
Algorithm 1 Representation of multi-objective Function

INPUT: \( U_l^C, U_l^{mm}, R_k^C, R_k^{mm}, S_k, CP_k^C \)

OUTPUT: \( MO_{kl} \)

1: for \( l \) = 1 to \( n \) do
2: \( W_l(t) = \alpha \sum_{t=t_1}^{t_2} U_l^C(t) + (1 - \alpha) \sum_{t=t_1}^{t_2} U_l^{mm}(t) \)
3: end for
4: for \( k \) = 1 to \( K \) do
5: \( L_k(t) = \alpha \sum_{t=t_1}^{t_2} R_k^C(t) + (1 - \alpha) \sum_{t=t_1}^{t_2} R_k^{mm}(t) \)
6: end for
7: for \( k \) = 1 to \( K \) do
8: for \( l \) = 1 to \( n \) do
9: if \( (\sum_{k=1}^{K} L_k(t), x_{kl} \leq W_l(t)) \) then
10: \( RU_{kl} = \frac{L_k(t)}{W_l(t)} \times 100 \)
11: end if
12: \( CO_{kl} = CO_{kl}^C + CO_{kl}^{mm} \)
13: \( CO_{kl}^{LD} = \frac{LD_{kl} \times CO_{kl}^{LD}}{\tau_3} \)
14: Total Cost \( TCO_{kl} = CO_{kl} + CO_{kl}^{LD} \)
15: Function \( MO_{kl} = \Delta_1 RU_{kl} + \Delta_2 TCO_{kl} \)
16: end for
17: end for
```
Metaheuristics – Accelerated Particle Swarm Optimization

- **PSO**
  - Designed based on the behavior of bird or fish schooling in nature
  - Searches the space of an objective function by adjusting the position of the particles in a quasi-stochastic manner
  - Finds an optimal particle based on two parameters: Velocity and Position

\[
v_i(t+1) = v_i(t) + \alpha \varepsilon_1 (G^* - X_i(t)) + \beta \varepsilon_2 (X^*(t) - X_i(t))
\]

\[
X_i(t+1) = X_i(t) + v_i(t+1)
\]

- **APSO**
  - No compelling to use the individual best unless the optimization problem is highly multimodal or nonlinear
  - APSO uses the global best location of a particle to accelerate the convergence speed of the problem

\[
v_i(t+1) = v_i(t) + \alpha \varepsilon + \beta (G^* - X_i(t))
\]

\[
X_i(t+1) = (1 - \beta)X_i(t) + \beta G^* + \alpha \varepsilon
\]
Quality of Service (QoS) vs Experience (QoE)

- **QoS** - refers to the overall features of system services which help to meet the stated and implied needs of the end users [ITU]
  - QoS drives through an agreement between user and provider that strongly monitors technical attributes of system services
  - Cost, service delivery deadline, packet loss ratio, jitter, throughput, etc.
  - E.g. Downloading a particular file in max 5 min time
- **QoE** is the total acceptability of a service that is determined by subjective perception of the end users [ITU]
  - Encapsulates user’s requirement, intentions and perceptions while provisioning system services
  - E.g. 2 users require that file in 3 min and 7 min respectively
- End users perceived QoE can degrade the acceptability of a service greatly even when the proper QoS is maintained
Quality of Experience (QoE)-aware Placement of Applications in Fog Computing Environments

- Fuzzy logic based approach that prioritizes different application placement requests
  - Based on the user expectations (*Rating of Expectation*)
- Fuzzy logic based approach that classifies Fog computational instances
  - Based on current status of the instances (*Capacity Class Score*)
- A linearly optimized mapping of application placement requests to Fog computing instances
  - To ensure maximized QoE-gain of the user

[Mahmud et al, JPDC 2019]
System model

• Fog-enabled IoT applications are divided into multiple interconnected Application Modules
  – E.g. Client Module and Main Application Module
• Fog nodes are organized in layers
  – Fog Gateway Nodes (FGN) and Fog Computational Nodes (FCN)
  – Provide RESTful services for querying and provisioning computational facilities
Calculating Rating of Expectation (ROE)

- Each application has certain parameters
- Range and unit of the parameters are not the same
  - So normalized to the interval \([-1, 1]\)
  - Parameter \(x\) values in range \([\alpha_x, \beta_x]\).
- Parameters and fuzzy sets
  - Access rate: \(\text{Ar} \in \{\text{Slow, Normal, Fast}\}\)
  - Required resources: \(\text{Rr} \in \{\text{Small, Regular, Large}\}\)
  - Processing time: \(\text{Pt} \in \{\text{Stringent, Moderate, Flexible}\}\).
- Fuzzy rules for ROE calculation
- Defuzzification using discrete center of gravity
Calculating Capacity Class Score (CCS)

• Parameters and fuzzy sets
  – Round trip time: $\text{Rtt} \in \{\text{Short, Typical, Lengthy}\}$
  – Resource availability: $\text{Ra} \in \{\text{Poor, Standard, Rich}\}$
  – Processing speed: $\text{Ps} \in \{\text{Least, Average, Intense}\}$.

• Discrete center of gravity

$$
\tau_{in} = \frac{\sum_{k=1}^{n} \mu_{C}(f_{in}^{k}) \times \Phi_{k}}{\sum_{k=1}^{n} \mu_{C}(f_{in}^{k})}.
$$
Mapping of applications to Fog instances

- Rating gain = ROE X CCS
- Total Rating Gain of the applications should be maximized

\[
\max \sum_{\forall a_m \in A_m} \sum_{\forall n \in N} \sum_{\forall i_n \in I_n} z_{i_n}^{a_m} (\eta_{a_m} \times \tau_{i_n})
\]

subject to,

\[
\sum_{\forall a_m \in A_m} z_{i_n}^{a_m} = 1; \forall n \in N, \forall i_n \in I_n
\]

\[
\delta_{a_m} \leq Q_\delta
\]

\[
\zeta_{a_m} \leq Q_\zeta
\]

\[
\rho_{a_m} \leq Q_\rho
\]

Maintain QoS - Service delivery time, service cost, and packet loss rate

<table>
<thead>
<tr>
<th>Simulation parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expectation metrics:</td>
<td></td>
</tr>
<tr>
<td>Access rate</td>
<td>2 - 10 per sec</td>
</tr>
<tr>
<td>Resource requirement</td>
<td>1 – 8 CPU cores</td>
</tr>
<tr>
<td>Processing time</td>
<td>30 – 120 ms</td>
</tr>
<tr>
<td>Status metrics:</td>
<td></td>
</tr>
<tr>
<td>Round trip time</td>
<td>100 – 600 ms</td>
</tr>
<tr>
<td>Resource availability</td>
<td>1 – 10 CPU cores</td>
</tr>
<tr>
<td>Processing speed</td>
<td>10 – 70 TIPS</td>
</tr>
<tr>
<td>Applications service delivery deadline</td>
<td>250 – 750 ms</td>
</tr>
<tr>
<td>Data signal loss rate of the network</td>
<td>3% – 5%</td>
</tr>
<tr>
<td>Service cost</td>
<td>0.1 – 0.15 $ per min</td>
</tr>
<tr>
<td>Number of accessible FCN per FGN</td>
<td>4 – 10</td>
</tr>
<tr>
<td>Data signal size</td>
<td>1000 – 2000 instructions</td>
</tr>
</tbody>
</table>
DPTO: A Deadline and Priority-aware Task Offloading in Fog Computing Framework Leveraging Multi-level Feedback Queueing

- **Task Priority 1:**
  - Class that supports the delay-sensitive tasks with hard deadlines
  - No negotiation on deadline
- **Task Priority 2:**
  - Tasks have intermediate priority with soft-deadline
  - Tasks meet their deadline with a negotiation and penalty
- **Task Priority 3:**
  - Lowest priority class that aims to support the resource-intensive tasks
  - No deadline
  - Mostly offloaded to the cloud

- **Goal:** To minimize the overall queueing waiting time and offloading time of the real-time tasks while meeting the deadline and resource

[Adhikari et al, IEEE IoTJ 2020]
Multi-level Feedback Queueing: Offloading

- Offloads the scheduled tasks (mainly hard-deadline and soft-deadline based tasks) on optimal fog nodes.
- Preferably deploy the without-deadline tasks on the centralized CDC.
- Offloading decision takes based on the CPU and memory capacity of the computing devices and requested resources by the applications.
HeRAFC: Heuristic Resource Allocation and Optimization in MultiFog-Cloud Environment

- The infrastructure and applications are modelled using graph theory
- FCI (Fog-Cloud Interface) is introduced, where a fog node can communicate with cloud and other fog nodes as well
- A task can be assigned to other fog nodes at multi-hop distance
- Heuristic approach is followed to achieve the objective of resource utilization maximization

[Dehury and Srirama, TCC 2020 (Under review)]
Fog Computing – Research Challenges - continued

• Process-driven Edge Computing in Mobile IoT
  [Mass et al, IoTJ 2019; CASA 2018; Chang et al, CSUR 2016]
Fog Computing – Research Challenges - continued

• Mobility also becomes critical in Fog computing [Mass et al, IoTJ 2019]

• STEP-ONE : Simulated Testbed for Edge Processes based on the Opportunistic Network Emulator [Mass et al, JSS 2020 (under review)]
  – Extended the ONE simulator to simulate the Fog computing mobility aspects
  – Process execution based on Flowable BPMS
STEP-ONE
Future Research Scope

• Standards-based dynamic deployment and management of fog applications
  – Deployment should consider heterogeneity, mobility aspects, multiple ownership (interoperability) etc.
  – Cloud computing community addressed similar problem through OASIS TOSCA
  – Extending TOSCA to support fog-based applications
  – Develop relevant orchestrators

• Efficient algorithms for self-adaptive resource provisioning and QoS-aware application module scheduling in fog topology

• Reliable and data-locality preserving real-time data processing in fog environments
  – Controlling the dataflow of distributed data processing applications across the multi-layer fog
  – Restructure the IoT applications into dynamically composable multi-stage data pipelines
  – Actor programming model and serverless functions edge analytics tasks
  – Apache Nifi data pipelines for task composition

• IoT case studies in domains such as smart city, smart healthcare, smart agriculture etc.
Next lecture

- Students present their projects
  - Full team should be present
- 15 min for each team
- Backed by slides
- Submit your source code and reports
References