# Vertical Workflows: Service Orchestration Across Cloud & Edge Resources

Omer Rana[1], Manjerhussain Shaikh[1], Muhammad Ali[2], Ashiq Anjum[2], Luiz Bittencourt[3]
[1]*School of Computer Science & Informatics, Cardiff University, UK*
[2] *University of Derby, UK*
[3] *University of Campinas, Brazil*
*contact author:* `ranaof@cardiff.ac.uk`

***Abstract*—Currently devices used for data capture often differ from those that are used to subsequently carry out analysis on such data. Many Internet of Things (IoT) applications today involve data capture from sensors that are *close* to the phenomenon being measured, with such data subsequently being transmitted to Cloud data centers for analysis and storage. Increasing availability of storage and processing devices closer to the data capture device, perhaps over a one-hop network connection or even directly connected to the IoT device itself, requires more efficient allocation of processing across such edge devices and data centers. We refer to these as "vertical workflows" – i.e. workflows which are enacted across resources that can vary in: (i) type and behaviour; (ii) processing and storage capacity; (iii) latency and security profiles. Understanding how a workflow pipeline can be enacted across these resource types is outlined, motivated through two scenarios. The overall objective considered is the completion of the workflow within some deadline constraint, but with flexibility on where data processing is carried out.**

***Keywords*-Sensor Applications, Cloud Computing, Fog & Edge Resource Allocation**

## I. Introduction

Internet of Things (IoT) is the network of physical objects that make use of embedded systems to communicate and sense or interact with their internal states or the external environment. Although estimates vary according to the survey being considered, it is generally expected that by the year 2025, 80 billion devices would be connected, and the amount of digital data created would reach 180 zettabytes[1]. With such increases in the number of devices and data volumes, it would no longer be sustainable to use cloud as the centralised server for computing and storage, as it is estimated that bandwidth is not likely to grow exponentially to support this increase in demand. Furthermore, many IoT and *edge* applications (e.g. autonomous cars) would require much lower latency than what is offered by a cloud data center today to operate efficiently. Fog computing is a paradigm that extends capabilities of cloud computing to the edge of the network, thus offering key benefits of cloud computing such as scalable compute and storage, minus limitations such as latency constraints. Fog computing provides an ideal paradigm for supporting real-time applications (such as data stream processing), as such applications have a shorter resource lifetime and hence require high speed response at near-edge servers, reducing communication delay to few milliseconds, rather than several hundred milliseconds when using cloud data centers. In addition, the use of fog computing enables offloading some of the computation-intensive processing on the user's device to (one or more) edge server(s), making application processing less dependent on device capability. Fog computing paradigm has certain distinct characteristics such as: (a) low latency data processing, (b) location awareness, (c) ability to handle high data volume, (d) user mobility, and support for (e) network heterogeneity.

However, Fog resources on their own are not enough to support application execution, as they may not have access to data sources or the required computational capacity to execute all the software components that make up an application. Understanding what should be executed on Fog resources compared to the data centre remains an important challenge, and the basis of this paper. We consider, in particular, the execution of a pipelined workflow on such cloud & fog resources. In pipeline processing, a succession of computational tasks (also called stages) is applied to a chain of input data elements in some order. Unlike iterative execution where each data element would have to go through the entire sequence of tasks before the next data element can be processed, in pipeline processing the next data element enters into the pipeline as soon as its predecessor has been processed by the first stage. Consequently, the overall execution time of the entire set of data elements is reduced by this concurrent and overlapped execution (as outlined in [2]). This type of constrained workflow is particularly important in applications which need to process a data stream (i.e. where data is continuously generated from a source). In a streaming pipeline data is continuously generated with either fixed or variable frequency, and processing involves applying one or more *operators* over a time or sample window of data (which is also application dependent). The raw data is often not archived, only the results of the analysis are stored. Processing of stream data can also trigger *events* that may subsequently *trigger* additional actions.

Enacting stream processing operations of this kind has been investigated extensively in the cloud computing com-

[1]https://www.forbes.com/sites/michaelkanellos/2016/03/03/152000-smart-devices-every-minute-in-2025-idcoutlines-the-future-of-smart-things/

munity, a good survey can be found in Ranjan [6]. This generally involves buffering a data stream, splitting the stream into a time/sample window and forwarding this to a cloud-hosted processing engine (such as Apache Kafka, Samza, etc). Other efforts have focused on creating specialist data structures that can improve the performance of continuous queries that can be applied to a data stream [7]. Amazon Kinesis provides an alternative approach that first divides a data stream and subseqently forwards this for analysis to a *lambda* function hosted at the nearest data center. Most of these approaches do not take account of the latency involved in transferring the data stream to the processing platform, and only measure performance of processing the stream at the data center. This can be restrictive for more realistic IoT applications, as the delay in transferring to the data center (from the capture point) can be significant. We use two application scenarios in section II to motivate the enactment of such workflow pipelines. Each application is analysed for using a combination of edge and cloud resources, and in particular how processing can be carried out across a combination of these two resources.

## II. Application Scenarios

We consider two application scenarios that make use of a stream processing pipeline. Both of these involve use of IoT resources to capture data, edge/fog resources to perform initial processing of the data, and a cloud data center for more computationally intensive jobs.

### A. Video Processing

Traditionally video data from data sources (cameras) is transferred to the cloud where data analysis takes place – ranging from archiving of the video feed, indexing frames, or supporting more complex operations such as object detection & recognition [8], as shown in Figure 1. The traditional model can suffer from high latency and unpredictable network bandwidth use, as all the data has to be transferred to the cloud for analytics. Two approaches to video analytics are (i) centralized, and (ii) use of a distributed architecture [4]. In a centralized approach data from video cameras are routed to a cloud platform for analysis, whereas in a distributed architecture part of the analytics can be performed near the data source and partly on the centralized cloud. Existing intelligent video analytics (IVS) systems are mostly based on a centralized approach and assume the video data to be readily available in proximity to where analytics takes place. In reality, the video data has to be moved through several network hops to reach the destination where it is stored and analyzed. Data transfer and subsequent processing can incur an initial overhead (based on the size of data being transferred). This data also has a *low-value density* [9], [4] – a term which expresses the usefulness of the data to support subsequent analysis. The value of a data

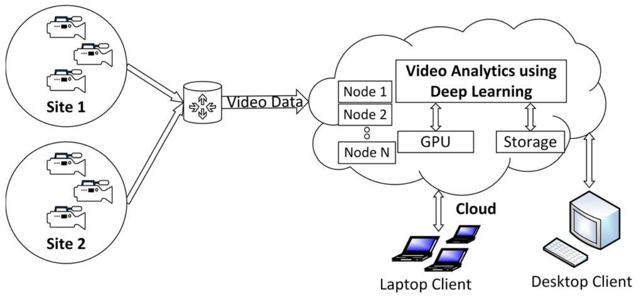


Figure 1: Traditional Cloud based Video Stream Analytics

is assessed on a subjective scale, and depends on the domain and the analytics problem.

The video analytics pipeline is illustrated in Figure 2a and consists of multiple stages. By default, all of the stages of the pipeline are executed on the cloud. These stages are obtained by decomposing the video analytics problem into parts which could be distributed among resources. As video data is submitted to the system, the first stage involves loading/decoding of the input data stream and motion detection (S1), consisting of a number of frames received at different rates from video sources which are then passed to a motion detector, where consecutive frames are analysed to detect any change. Frames with detected motion are subsequently passed to a frame enhancement stage (S2) where frame is analysed using image processing algorithms (e.g. histogram equilization, smoothing and or scaling) as a precursor step to an object detection stage. The enhanced frames are then fed to an object detector stage (S3) which involves detecting the presence of an object in the frame (S3), followed by an object recognition stage which labels the object (S4). The edge enhanced pipeline for video analytics is shown in Figure 2b, video pipeline stages are distributed among available resources of edge, cloudlet and cloud. The strategy for stage deployement can directly affect the performance of the system and depends on resource availability, time taken by each stage and compute power of each resource deployed. Stages are classified as i) basic stages and ii) machine learning stages. As shown in Figure 2 basic stages 1 and 2 are deployed on the edge nodes and machine learning stages 3 and stage 4 on cloudlet and cloud respectively. As an edge device can be resource constrainted, only the basic stages are deployed here. Filtering of low-value density data can occur at stages 2 & 3, i.e. frames that do not contain motion/ object can be discarded at these stages, limiting the size of data that needs to be trasferred to to the centralized cloud – saving bandwidth, storage and computational resources.

Several challenges have to be overcome to implement the video analytics pipeline on real hardware, such as identifying stages which should be hosted close to the data source and those on the cloud. Each stage takes a certain time to process, depending on the computational complexity of

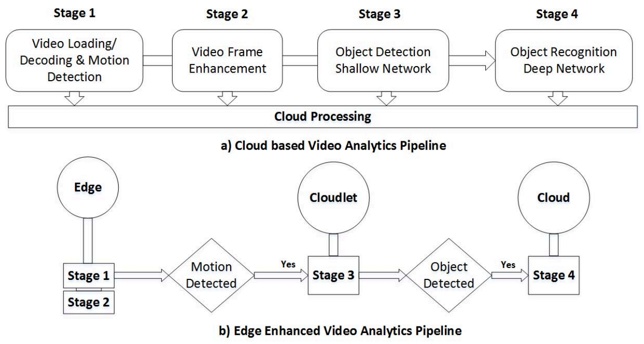

Stage 1
Stage 2
Stage 3
Stage 4
Video Loading/ Decoding & Motion Detection
Video Frame Enhancement
Object Detection Shallow Network
Object Recognition Deep Network
Cloud Processing
a) Cloud based Video Analytics Pipeline
Edge
Cloudlet
Cloud
Stage 1
Stage 2
Motion Detected
Yes
Stage 3
Object Detected
Yes
Stage 4
b) Edge Enhanced Video Analytics Pipeline

Figure 2: Video Analytics Pipeline

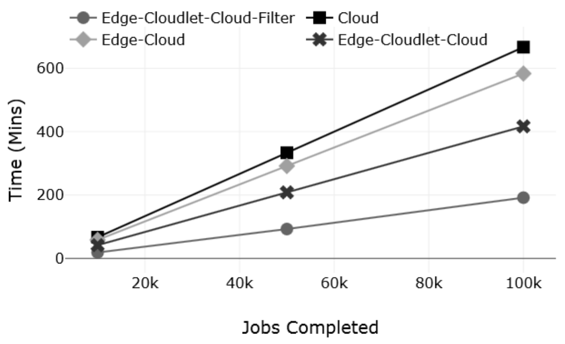


Figure 3: Processing time of 10K, 50K, 100K video streaming jobs – experiment infrastructure described in [4]

the stage and the resource on which it is executed. If data production rate exceeds processing capacity, then additional data has to be buffered at each resource. However, if the waiting time exceeds a threshold, the next resource in the pipeline may sit idle waiting for data to arrive. To overcome this and to maximize the use of resources, each resource is assigned a deadline to process the data in its current buffer, after which it sends the remaining data to the next stage in the pipeline. Since the stages are being distributed on over a network, they may suffer from network issues, that is stage 3 executing before stage 2. To overcome this, a mechanism to serialize the execution of stages has to be developed.

Experimental setup to distribute the pipeline stages over physical resources was realized using software defined networking (SDN) in a Simulator. Four different configurations of SDN were used: i) Cloud only (CO); ii) Edge-Cloud (EC); ii) Edge-Cloudlet-Cloud (ECC); and iii) Edge-Cloudlet-Cloud-Filter (ECCF), to test the performance of the system. Stages S1-S4 as shown in Figure 2 were deployed among available resources in each of the configuration. In CO configuration all stages were deployed on the cloud, in EC case, stages 1 and 2 were deployed on Edge and stages 3 and 4 on the cloud. In ECC case, stages 1 and 2 were deployed on edge, stage 3 on cloudlet and stage 4 on the cloud. In ECCF stages were deployed as in ECC case but here we also filter streams based on object detection, streams with no objects are discarded at the cloudlet. For each of these configuration, we consider time to process 10K, 50K and 100K video frames (jobs). Results in Figure 3 show that the time to process 10K jobs in all of the configurations are equivalent, however as more streams are produced the different configurations start to diverge. To process 100K jobs, CO configuration takes $\approx$ 700mins, where as all of the other configurations take less time. In ECC case, time to process 100K jobs is $\approx$ 400mins, i.e. ECC case is more efficient for processing 100K jobs compared to CO configuration. In ECCF case which is essentially the same as ECC but with an object filter at the cloudlet node, we can see further reduction in time to process 100k jobs, due to a reduction in the size of data that needs to be analysed.

## *B. Smart Traffic Management*

The second scenario we consider involves supporting traffic management taking account of sensor data that has been acquired in real time. A traffic management scenario is used to illustrate how different types of fixed and mobile input sensors may be used as road side units (making use of "dedicated short range communication" (DSRC)) to detect traffic conditions, which subsequently regulate the flow of traffic. Autonomous Vehicles involve communication with such road side units and an in-vehicle diagnostics gateway that have transceivers and transponders. These units are used to acquire the necessary traffic information, such as instance time, speed and the location of the vehicle. This data is then used to approximate the Start of Congestion (SoC) and Travel Time (TT) of vehicles.

In *traditional* IoT based smart traffic management, sensors send input parameter readings to a Cloud node which offers large compute and storage capability but incurs a higher latency. Using data that could be delayed in transit, control actuators may be responding to sensor readings which are a few seconds old, and traffic conditions may have changed. Hence fog computing offers a low latency traffic management solution which can respond to traffic events in time to control flow of traffic, avoid hazards and accidents, and deliver better experience to drivers. Our proposed fog computing based traffic management system makes use of the following key components, illustrated in Figure 4: (i) **Input Sensors** which detect traffic events and trigger computation on respective traffic analytics modules such as a Client and Traffic Density module on the fog device. In the first instance, we consider a single sensor "PPL". Such sensors are installed on roads at a predefined, fixed distance from each other. Each sensor continuously records all traffic parameters. If the sensor detects any predefined event, then the sensor will send the data to a fog device for analysis. Some examples of input sensors are vehicle & density detection, flow monitoring & congestion, wind vane, and incident detection (ii) **Global Display Actuators** receive updated instructions from a controller or a router fog device and adjusts the associated parameter values such as speed, lane closure signs, automated road or lane barriers, vehicle route displays etc; (iii) **Controller Fog device** provides compute, storage and network capability, and receives input from sensors and sends updated control instructions to an actuator. Controller Fog device is responsible for maintaining and managing local state, but they are also connected to a router fog device to support edgeward placement if necessary; (iv) **Router Fog device** is a network router connected to the controller fog device to channel data using a variety of different communication protocols. Such devices can enable co-existence of multiple communication channels, using IEEE 802.11p, Bluetooth, Wifi-Direct, etc, enabling switching between these channels based on network availability/quality and congestion. A router fog device may also have specialist computational capability to support the controller fog device; (v) **Cloud device** represents the data center and it is also responsible for maintaining global application state. In the edgeward placement strategy, application modules are deployed close to the sensors and actuators at the network edge. However, a network edge device may not have the computational capability to host all operators required by the application. In such a case, an attempt is made to place remaining operators on an available fog device. In the cloud-only placement, all application modules are executed in the cloud, indicating that the *sense-process-actuate* loop will be implemented by having sensors send input data to the cloud for processing, aligning with strategies used in the iFogSim simulator [12].

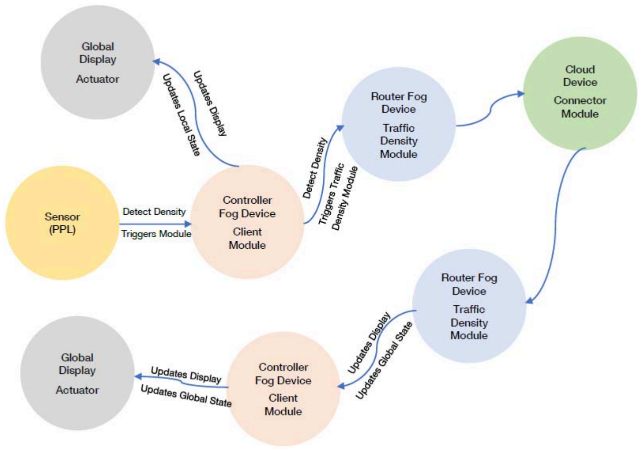


Figure 4: Traffic Management: Integrating Fog & Cloud

Application state, in this instance, indicates how an input and output is processed and may be local or global. In a local state input sensor and control actuators are both connected to the same controller device and this controller serves as the fog device. Alternatively, global state involves the control actuators being connected to a different controller fog device than the controller device to which the sensor is connected. In such a case, a cloud device (connector module) is used to support communication. Figure 4 illustrates the workflow pipeline showing how this state information is exchanged between the components that make up the traffic management system.

Both the video processing and traffic management system have parallels: (i) the data analysis pipleline comprises of a number of stages, and it is often preferred to undertake initial processing close to data generation – left hand side of Figure 4; (ii) processing is dependent on the device being considered and its current capacity and availability (i.e. fog controller or router). If fog resources are unavailable, a cloud data center is made use of; (iii) both scenarios involve processing of dynamically generated, real time data, that must be processed within some deadline constraints. The latency of data transfer is therefore significant and influences how pipeline stages get mapped to particular resources in the system (e.g. fog or cloud).

### C. *Simulation*

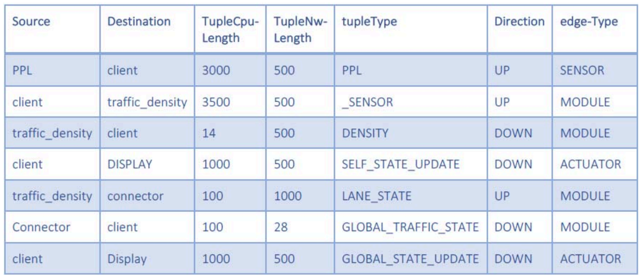
The traffic management scenario has been simulated using iFogSim through three specific modules: (i) Client, (ii) Traffic Density and (iii) Connector modules. The Client module takes a PPL tuple from the sensor, performs computation as a fog device and then passes SELF_STATE_UPDATE tuple to the display control actuator. The client module takes _SENSOR tuple as input and passes it to the traffic density module and is the most critical module which co-ordinates between sensors, actuators and manages local state. The Traffic Density module takes a _SENSOR tuple from client and passes the computed DENSITY to the Client Module running on the controller fog device. Traffic Density module runs on the router fog device and is invoked during an edgeward placement strategy, when the controller fog device is unavailable. The Connector module takes a LANE_STATE tuple from traffic density module running on the router fog device and passes GLOBAL_TRAFFIC_STATE tuple to the client module running on the controller fog devices, which in turn passes GLOBAL_STATE_UPDATE to a display. Table I identifies the parameters used to set up the simulation.

## III. Results

We evaluate the benefit of using an edgeward placement of services within the Smart Traffic scenario based on two criteria: (i) latency benefits; (ii) cost/revenue benefits.

### A. *Latency Measurements*

The smart traffic pipeline is executed multiple times by varying properties of components within the fog and cloud components. Two scenarios are created – a cloud only and



| Source | Destination | TupleCpu-Length | TupleNw-Length | tupleType | Direction | edge-Type |
|---|---|---|---|---|---|---|
| PPL | client | 3000 | 500 | PPL | UP | SENSOR |
| client | traffic_density | 3500 | 500 | _SENSOR | UP | MODULE |
| traffic_density | client | 14 | 500 | DENSITY | DOWN | MODULE |
| client | DISPLAY | 1000 | 500 | SELF_STATE_UPDATE | DOWN | ACTUATOR |
| traffic_density | connector | 100 | 1000 | LANE_STATE | UP | MODULE |
| Connector | client | 100 | 28 | GLOBAL_TRAFFIC_STATE | DOWN | MODULE |
| client | Display | 1000 | 500 | GLOBAL_STATE_UPDATE | DOWN | ACTUATOR |

Table I: Traffic Management Scenario. TupleCPU is in MIPS, and TupeNW (network) in Mbps

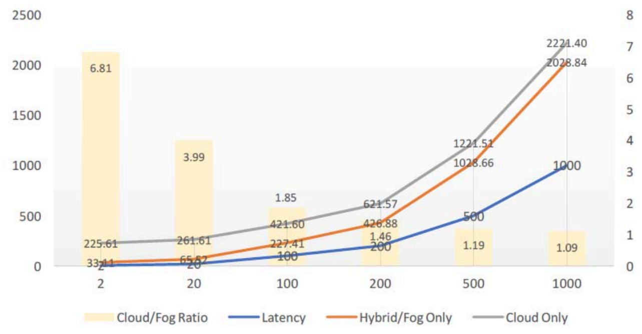


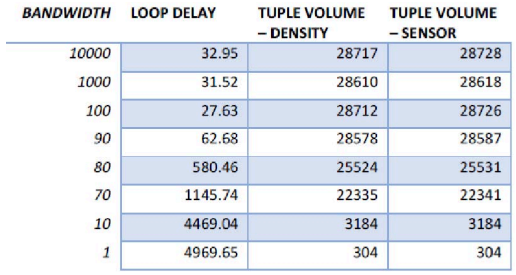
Figure 5: Impact of latency on application execution – latency (x-axis) in seconds; loop delay (y-axis) in seconds

fog (hybrid, i.e. fog & cloud) to understand the impact of latency on the pipeline execution time (referred to as *loop delay*). Figure 5 provides a performance comparison between these different placement strategies.

Hybrid/fog-only and cloud-only nodes react differently to the same incremental change in latency. While we observe a gradual increase in latency in both scenarios, for a 500s latency, the *loop delay* in figure 5) for hybrid or fog only increases by 61.25 times, while loop delays for cloud only increase by 9.84 times. Hence as latency changes, fog computing may lose its benefit of low-latency, and faster processing over cloud infrastructure.

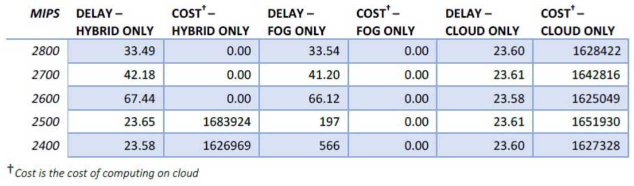
We also consider the impact of modifying the bandwidth on the overall execution time of the application and the size of data (number of tuples) that can be exchanged and processed. As outlined in table II, as bandwidth increases, the loop delay reduces (as expected), allowing higher number of tuples to be processed by density and sensors modules. For this simulation, we have considered a hybrid-scenario instead of fog or cloud only, as such changes do not directly impact cloud based systems.

We also investigate the impact of changing processing capacity (in terms of MIPS rating of CPUs on the devices) on the overall loop delay and the cost of using a cloud data center. We created three scenarios: cloud-only, hybrid and fog-only. To create the fog only scenario we disabled the connector module manually and reduced MIPS values. From



| *BANDWIDTH* | LOOP DELAY | TUPLE VOLUME – DENSITY | TUPLE VOLUME – SENSOR |
|---|---|---|---|
| *10000* | 32.95 | 28717 | 28728 |
| *1000* | 31.52 | 28610 | 28618 |
| *100* | 27.63 | 28712 | 28726 |
| *90* | 62.68 | 28578 | 28587 |
| *80* | 580.46 | 25524 | 25531 |
| *70* | 1145.74 | 22335 | 22341 |
| *10* | 4469.04 | 3184 | 3184 |
| *1* | 4969.65 | 304 | 304 |

Table II: Impact of bandwidth on delay & data size (tuples) exchanged.



| *MIPS* | DELAY – HYBRID ONLY | COST† – HYBRID ONLY | DELAY – FOG ONLY | COST† – FOG ONLY | DELAY – CLOUD ONLY | COST† – CLOUD ONLY |
|---|---|---|---|---|---|---|
| *2800* | 33.49 | 0.00 | 33.54 | 0.00 | 23.60 | 1628422 |
| *2700* | 42.18 | 0.00 | 41.20 | 0.00 | 23.61 | 1642816 |
| *2600* | 67.44 | 0.00 | 66.12 | 0.00 | 23.58 | 1625049 |
| *2500* | 23.65 | 1683924 | 197 | 0.00 | 23.61 | 1651930 |
| *2400* | 23.58 | 1626969 | 566 | 0.00 | 23.60 | 1627328 |

† *Cost is the cost of computing on cloud*

Table III: Impact of MIPs change & Cost. Delay refers to the execution time (*loop delay*)

table III we can observe that the loop delay for a hybrid model increases steadily with a decrease in MIPS. The loop delay values for hybrid models are similar to loop delay values for fog only models, when MIPS values are decreased from 2800 to 2600. This can be attributed to all processing taking place on fog devices only (even in the hybrid model). We further observe that the loop delay peaks at a MIPS value of 2600 and then reduces as processing moves to the cloud.

## B. Revenue

We investigate the impact of varying: MIPS, memory and bandwidth on potential revenue generated by fog nodes. The total revenue is the sum of revenues for compute, memory and network resources consumed by users making use of fog nodes. Our cost analysis is based on energy consumption using the *UK 2017* tariff, where fixed, non-commercial, per unit cost of 14.31 pence for Kwh is used. We have limited the cost to energy consumption and have excluded all other costs such as rentals, network charges, etc. All capital expenditure (CAPEX) costs such as hardware purchases and any other procurements have been excluded in this analysis. In table IV, we determine energy consumption, MIPS, memory and bandwidth at different energy unit tariff (first column). Total cost is calculated by multiplying energy consumption with per unit rate of energy, i.e. 14.31p. Total revenue in the last column is a sum of revenue for MIPS, memory and bandwidth, and is influenced by unit rate change. In tables IV and V the break-even point when considering a cost based model with a per unit rate of 0.67p/kWh is provided.

In a user based revenue model, the number of subscribers needed to meet the revenue threshold is computed by varying

| *UNIT RATE* | ENERGY CONSUMPTION | ENERGY COST | MIPS REVENUE | MEMORY REVENUE | BANDWIDTH REVENUE | TOTAL REVENUE |
|---|---|---|---|---|---|---|
| *0.001* | 27892764 | 399145459 | 205448 | 3903 | 390292 | 599643 |
| *0.005* | 27892975 | 399148468 | 1027272 | 19460 | 1946040 | 2992772 |
| *0.01* | 27892975 | 399148468 | 2055024 | 39003 | 3900280 | 5994307 |
| *0.015* | 27892789 | 399145804 | 3081962 | 58493 | 5849310 | 8989765 |
| *0.02* | 27893606 | 399157503 | 4112178 | 78084 | 7808440 | 11998702 |
| *0.025* | 27891993 | 399134416 | 5132668 | 97452 | 9745150 | 14975269 |
| *0.03* | 27892829 | 399146385 | 6164026 | 116992 | 11699220 | 17980238 |
| *0.05* | 27893259 | 399152535 | 10272555 | 194966 | 19496600 | 29964121 |
| *0.10* | 27893259 | 399152535 | 20555699 | 390982 | 39098200 | 60044881 |
| *0.20* | 27893303 | 399153161 | 41109969 | 779180 | 77918000 | 119807149 |
| *0.30* | 27892525 | 399142026 | 61625149 | 1170576 | 117057600 | 179853325 |
| *0.40* | 27892671 | 399144118 | 82174468 | 1563568 | 156356800 | 240094836 |
| *0.50* | 27892612 | 399143278 | 102719977 | 1949150 | 194915000 | 299584127 |
| *0.60* | 27892422 | 399140563 | 123228575 | 2344152 | 234415200 | 359987927 |
| *0.65* | 27892881 | 399147128 | 133544787 | 2536326 | 253632600 | 389713713 |
| *0.70* | 27892881 | 399147128 | 143845100 | 2727550 | 272755000 | 419327650 |

Table IV: Revenue analysis at different unit energy costs

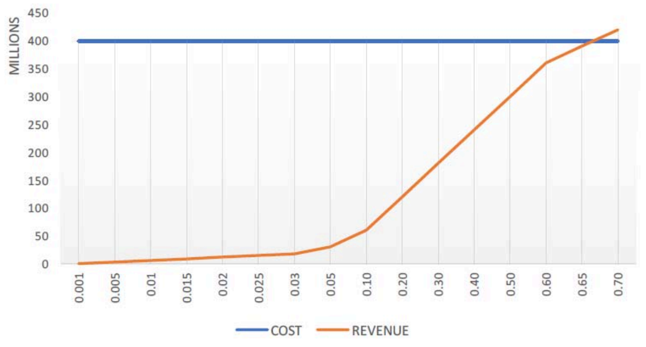


Table V: Cost/revenue breakeven analysis

| *USERS* | UNIT RATE | ENERGY CONSUMPTION | TOTAL COST | MIPS REVENUE | MEMORY REVENUE | BANDWIDTH REVENUE | TOTAL REVENUE |
|---|---|---|---|---|---|---|---|
| *3* | 0.40 | 19495739 | 278984032 | 22037331 | 421816 | 42181600 | 64640747 |
| *6* | 0.40 | 22266128 | 318628299 | 40873254 | 800328 | 80032800 | 121706382 |
| *9* | 0.40 | 25175080 | 360255400 | 66064845 | 1171648 | 117164800 | 184401293 |
| *12* | 0.40 | 27893017 | 399149069 | 82204597 | 1565728 | 156572800 | 240343125 |
| *16* | 0.40 | 32469702 | 464641440 | 109409645 | 2078392 | 207839200 | 319327237 |
| *20* | 0.40 | 37987735 | 543604492 | 249129639 | 3416584 | 341658400 | 594204623 |
| *24* | 0.40 | 41518812 | 594134193 | 266812676 | 3811776 | 381177600 | 651802052 |

Table VI: Changes with user numbers

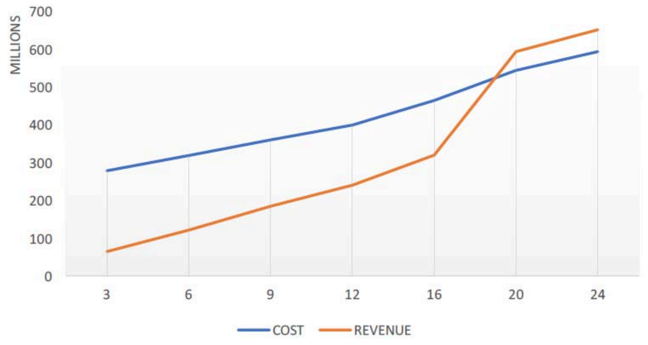


Figure 6: Cost vs. revenue analysis

the number of areas and number of controllers per area. In this simulation revenue is computed using MIPS, memory and bandwidth, for a different number of users and a fixed unit rate of 0.40p/Kwh. Energy consumption is computed for different number of users and the total cost is calculated as a product of energy consumption and per unit charge for energy usage (excluding hardware costs, rentals, network usages etc).

We observe (from table VI) that the values for parameters including energy consumption, MIPS, memory and bandwidth increase steadily with incremental changes in number of users. As the unit rate is fixed at 0.40, revenues are only impacted by the number of users, and device efficiency would also significantly affect the break-even point (figure 6). Hence a device with a lower energy consumption would likely reach the break-even point faster than other less efficient devices.

As discussed above, we considered three critical parameters to characterise performance of fog nodes: latency, MIPS and bandwidth, and associate these with application properties: tuple volume, loop delay and cost of execution. For cost of execution we have only considered cloud-based execution and assumed that the fog devices are owned and controlled directly by a user (i.e. fog device do not have any cost of execution). Overall, our analysis shows that: (i) latency directly impacts loop delay hence increase in latency will increase loop delays. With an increase in latency the ratio of cloud to fog loop delay reduces, which means that impact of latency on fog devices is significantly higher than those on cloud nodes. If a computing environment does not change, then the cloud to fog ratio approaches 1:1, at which stage, fog computing loses its advantage of low latency; (ii) bandwidth inversely impacts loop delay hence a decrease in bandwidth results in increase in loop delay. Bandwidth also directly impacts tuple volume that can be processed, which means that a decrease in bandwidth results in a lower volume of tuples that can be processed by fog nodes, leading to a queueing of tuples, which in turn can result in an increase in latency; (iii) a decrease in MIPS rating of fog nodes inversely impacts processing capacity, resulting in an increase in loop delay for fog devices. We have adjusted MIPS only in fog devices to assess impact of different computing capability, and we assume that the cloud has infinite resources – hence we can exclude impact of MIPS change in cloud. However as fog devices migrate processing to cloud resources, this would result in users experiencing higher latency.

These experiments can be used as a basis for resource planning for executing pipeline workflows. A system developer can assess likely impact on performance (loop delay) by changing device capacity (MIPs rating) and available memory, and assessing potential impact of bandwidth change.

## IV. Related Work

Enacting workflow across cloud resources has received significant attention in both scientific and business applications. Generally, this involves splitting an application into a number of sub-components/ stages, and submitting each stage to one or more cloud system(s). However, enactment across an infrastructure that combines IoT, edge and cloud systems remains limited. Two key challenges may be considered in this context: (i) how services within a workflow can be distributed across edge and cloud resources, and how the subsequent allocation can be adapted [1]; (ii) how data can be staged across workflow stages to improve overall execution time. Our focus in this work is particularly on (i).

Bonomi et al. [5] introduce a layered model combining application execution across IoT and (potentially multiple) cloud providers. They describe the requirement for a fog computing framework to utilize multiple communication links and protocols, depending on how data is exchanged between the different layers (i.e. cloud to fog, fog to IoT, etc). Dastjerdi et al. [10] present a reference architecture for fog computing which follows a very similar structure. The reference architecture involves serving IoT requests using locally available resources at fog nodes, rather than at a centralised cloud data center. This is achieved using a Software-Defined Resource Management middleware, which provides coordination between different resources that make up the fog layer. A key focus in both of these efforts is the need to orchestrate resource allocation across fog and cloud resources, and preventing each layer acting in an autonomous manner. In the same way, Vaquero et al. [11] consider both centralized and decentralized architectures to realise fog nodes, introducing the notion of "edge cloud" made up of multiple IoT devices. Mach and Becvar [13] identify this as the "Fog Service Placement Problem", investigating offloading of services from the client's device to the cloud or edge resources. They assume all services are identical, and therefore do not take account of additional capacity made available at a cloud data center.

## V. Challenges

Given the application scenarios and their use of fog/egde computing capabilities, we identify a number of challenges that need to be considered for the future. These are categorised into two areas: (i) technical; (ii) economic.

**Challenges – Technical:**

- Security and data privacy is often identified as a benefit of using edge computing. However understanding what security operations to perform on edge resources to extend what is available within a data center needs to be more explicitly identifed. Similarly, keeping edge devices secure now becomes a concern for a user, requiring frequent checks on the type of firmware available on the device. Given the potential heterogeneity of edge devices, this challenge is likely to become more significant.
- Preserving data privacy would mean that a user can limit the type and range of data they share with a data center. This however necessitates suitable computational and storage capability to be available at edge nodes.
- Understanding how an application can be structured as a workflow pipeline is also a challenge. To make more effective use of the underlying fog and cloud infrastructure, it is also important to enable pipeline stages to migrate across different parts of the infrastructure. This is a key premise of "osmotic computing" [1], which identifies the use of container-hosted microservices that can be moved from fog devices to cloud data center (and vice versa). Understanding what services can be developed that enable multiple versions to co-exist, i.e. a service that can be deployed within a data center on a large-scale system and another that is a resource constrained (approximate) version of this service which could be deployed at the edge.

**Challenges – Economic:** Understanding suitable business and economic models that will encourage deployment of fog nodes remains a challenge. A key question that underpins this is a better understanding of who should manage and coordinate a fog-based infrastructure, i.e. should these be existing cloud and data center operators, telecomm. vendors & network operators interested in making use of their existing investments in mobile systems, or a new type of company that acts as a broker between these two? Much of this is also dependent on the potential number of applications that can make more effective use of edge infrastructure.

Should such an infrastructure be vendor specific or neutral – i.e. does each operator primarily extend their existing system to the network edge, or is there a need to consider more interoperation within these systems (similar the use of mobile data usage today)? Given this context, we identify the following challenges in deploying and managing fog nodes:
(i) Fog node operators should enable hosting of services on their infrastructure. Understanding the incentives for undertaking this remain to be clarified. For instance, would fog operators be telecomm. providers who offer mobile network infrastrucuture or cloud data center operators?
(ii) Alternatively, should cloud operators (e.g. Amazon AWS) form alliances with fog operators? Alternatively, should cloud operators manage their own telecomms. infrastructure in close proximity to users to enable deployment of fog nodes. This is similar to existing efforts in Content Distributed Networks (e.g. Akamai and Amazon CloudFront), where a server ensemble is used to process user requests in proximity to user/data location.

## VI. Conclusion

Internet of Things devices and applications that generate and consume data are scattered over the edge of the network. Thus, IoT can benefit from data processing at the edge, but the centralised cloud data centres also play a role due to their large computational capacity. The combination of processing at the edge, e.g. with fog computing, and in the cloud work on symbiosis to deliver better quality of service to IoT as well as to improve infrastructure utilisation.

In this paper we discussed how fog and cloud computing can be used to process *vertical workflows*, which have jobs that are suitable to run at the edge and in the cloud in a pipelined fashion. Two case studies were used to illustrate how this kind of workflow can be modelled and run in such a hybrid infrastructure. A smart traffic scenario was evaluated in terms of latency and cost/revenue benefits, suggesting that fog-cloud hybrid infrastructure is indeed suitable to run vertical workflows. Moreover, a set of challenges were identified, and discussed, illustrating many research opportunities in fog-cloud scenario for IoT. Enacting workflows in this manner implies availability of services which can be enacted on either edge or cloud resources. Depending on availability, background workload profile of a resource and QoS requirement of an application, it is possible therefore to migrate a service from a cloud data center to an edge resource (and vice versa). Understanding what can be migrated and when is another research challenge identified in this work.

**Acknowledgements:** Bittencourt's contribution was partially funded by the European Commission H2020 programme, grant no. 688941 (FUTEBOL), as well from the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through Brazilian National Research and Educational Network (RNP) and CTIC. LFB would like to thank CNPq and CAPES for the financial support.

## References

[1] M. Villari, M. Fazio, S. Dustdar, O. Rana and R. Ranjan, “Osmotic Computing: A New Paradigm for Edge/Cloud Integration,” in IEEE Cloud Computing, vol. 3, no. 6, pp. 76-83, 2016. doi:10.1109/MCC.2016.124

[2] R. Tolosana-Calasanz, J. A. Banares, O. F. Rana, “Autonomic streaming pipeline for scientific workflows.” Concurrency and Computation: Practice and Experience 23(16): 1868-1892 (2011). John Wiley.

[3] A. Cuzzocrea, G.Fortino and O. Rana, “Managing Data and Processes in Cloud-Enabled Large-Scale Sensor Networks: State-of-the-Art and Future Research Directions”, DPMSS workshop alongside CCGrid 2013, Delft, The Netherlands, pp 583-588, IEEE Computer Society Press.

[4] M. Ali, A. Anjum, M. Yaseen, A. Zamani, D. Balouek-Thomert, O. F. Rana, and M. Parashar, “Edge enhanced deep learning system for large-scale video stream analytics.” Proc. IEEE 2nd Int. Conf. on Fog and Edge Computing (ICFEC), Washington DC, USA, 1-3 May 2018.

[5] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, “Fog computing: a platform for internet of things and analytics.” In: Big data and internet of things: a roadmap for smart environments. Studies in computational intelligence 546: 169186. Springer, 2014.

[6] R. Ranjan, “Streaming Big Data Processing in Datacenter Clouds.” IEEE Cloud Computing 1(1): 78-83 (2014)

[7] Ze Deng, Xiaomin Wu, Lizhe Wang, Xiaodao Chen, Rajiv Ranjan, Albert Y. Zomaya, Dan Chen, “Parallel Processing of Dynamic Continuous Queries over Streaming Data Flows.” IEEE Trans. on Parallel Distrib. Syst. 26(3): 834-846 (2015)

[8] C. Regazzoni, A. Cavallaro, Y. Wu, J. Konrad, and A. Hampapur, “Video analytics for surveillance: Theory and practice [from the guest editors],” IEEE Signal Processing Magazine, vol. 27, no. 5, pp. 1617, 2010.

[9] A. Gandomi, M. Haider, “Beyond the hype: Big data concepts, methods, and analytics”, Int. Journal of Information Management Volume 35, Issue 2, April 2015, pp 137-144. Elsevier.

[10] A. Dastjerdi, H. Gupta, R. Calheiros, S. Ghosh, R. Buyya, “Fog computing: principles, architectures, and applications.” In: Internet of things: principles and paradigms, chap. 4, Morgan Kaufmann, 2014

[11] L. Vaquero, L. Rodero-Merino, “Finding your way in the Fog: Towards a Comprehensive Definition of Fog Computing.”, ACM SIGCOMM Comput Commun Rev 44(5):2732, 2014.

[12] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, R. Buyya, “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments”. Available at: https://arxiv.org/abs/1606.02007, June 2016.

[13] P. Mach and Z. Becvar, “Mobile Edge Computing: A Survey on Architecture & Computation Offloading”, IEEE Comms. Surveys Tutorials 19,3, pp 1628–1656, 2017.