VITTORIO SCARANO

Computation offloading to the Edge

VITTORIO SCARANO

LESSON PLAN

1. Cloud and Edge Computing
2. Fundamentals
3. Computation Offloading
4. Challenges
5. Application Scenarios
6. Opportunities and conclusions

MAP

WHAT IS CLOUD COMPUTING

According to NIST

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.""

- USA National Institute of Standards and Technology
- "Model"
- Resources: networks, servers, storage, application and services
- Minimal management effort/interaction

OFFLOADING COMPUTATION TO THE EDGE

- Massively diffused end devices perform computing everywhere and everyday.
- Problems: constrains by the battery and computational/memory resources.
- First solution: shift to perform computation offloading to the cloud, known as Mobile Cloud Computing.
- But . . . the cloud is usually far away from end devices
  - high latency (around 100ms), and low quality of experience
- Edge computing extends the cloud to the edge of the network, close to end users, bringing ultra-low latency and high bandwidth.

A NEW GAME WITH NEW RULES!

- Computation offloading in edge computing bears a superficial resemblance to Mobile Cloud Computing.
- Task placement is not only two options, i.e., either at local or in the cloud, but also possible on any edge node.
  - Moreover, Edge is usually a loosely coupled heterogeneous wireless networks: complicated decisions
- Code partitioning is complex: task execution is distributed among mobile devices, multiple edge nodes, and the cloud
- Balancing within the Edge with a standard deployment is not currently available.


ABC

MAP

VITTORIO SCARANO

According to ETSI

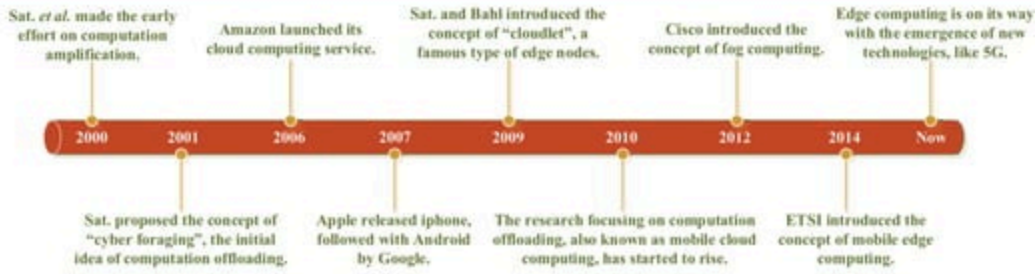
"Mobile Edge Computing provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers. The aim is to reduce latency, ensure highly efficient network operation and service delivery, and offer an improved user experience. .""

- ETSI: European Telecommunication Standard Institute
- Close to radio network
- Integrated in the 5G effort
- Improved user experience

EDGE VS. FOG COMPUTATIONS

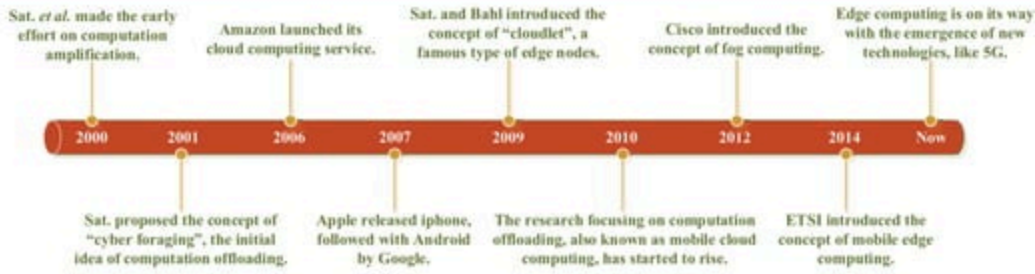
- Introduced by Cisco in 2012
- Fog computing highlights the characteristics, such as distributed processing, online analytics, interplay with the cloud, and network management.
- Difference: "While edge computing refers more specifically to the computational processes being done at or near the edge of a network, ...
- ... fog computing refers to the network connections between the edge devices and the cloud."
- Edge: architecture of computing at the edge
- Fog: uses edge computing and defines the network connection over edge devices, edge servers, and the cloud.

A TIMELINE OF COMPUTATION OFFLOADING - 1



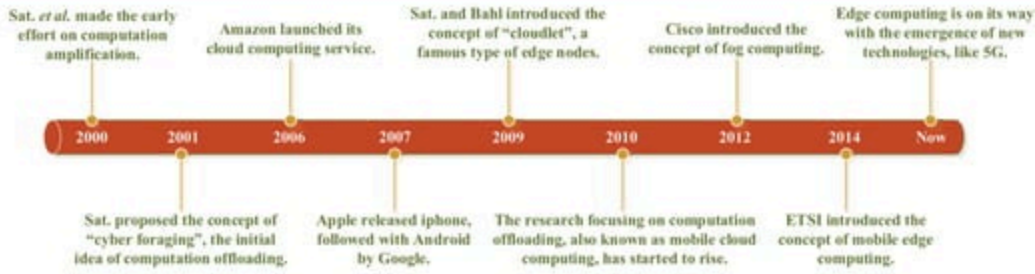
- Studies on how to improve the performance of mobile applications and save the energy of mobile clients by collaborative computing with remote servers.
- "Odyssey", a prototype as a proof of concept to support the adaptive execution of mobile applications.
- Parameters: CPU cycles, bandwidth, and battery power.

A TIMELINE OF COMPUTATION OFFLOADING - 2



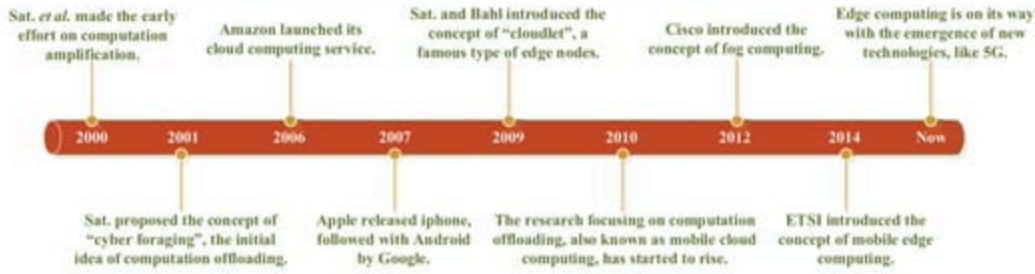
- Satyanarayanan characterized pervasive computing as the integration of distributed computing and mobile computing
- Cyber foraging: to augment the computing capability of a thin mobile device dynamically by leveraging the power of wired hardware infrastructure.
- Considered the origin of computation offloading.

A TIMELINE OF COMPUTATION OFFLOADING - 3



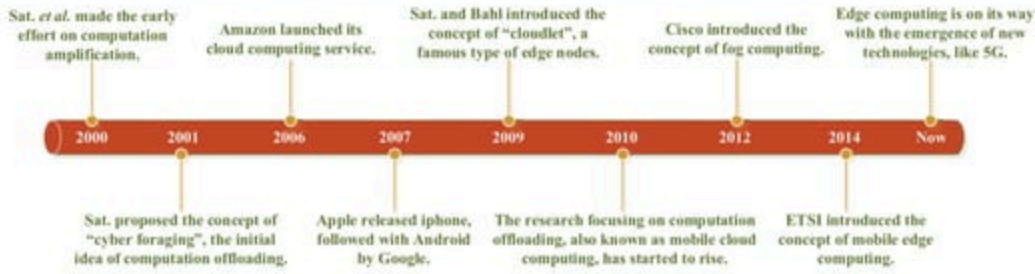
- Two computing paradigms: coud computing and mobile computing
- Amazon commercializes its elastic compute cloud (EC2) in 2006
- In 2007: the release of mobile OSs, like iOS and Android, on smartphones rich in applications and sensors come in people's life, and it leads to the prosperity of mobile computing.
- Many works: MAUI, Cuckoo, CloneCloud, ThinkAir, COMET

A TIMELINE OF COMPUTATION OFFLOADING - 4



- In 2009, the concept of "cloudlet": a small Data Center nearby mobile devices, which is a proof of concept for edge computing.
- A three-tier architecture consisting of mobile devices, cloudlets, and the cloud,
- Aims: "bring the cloud closer"

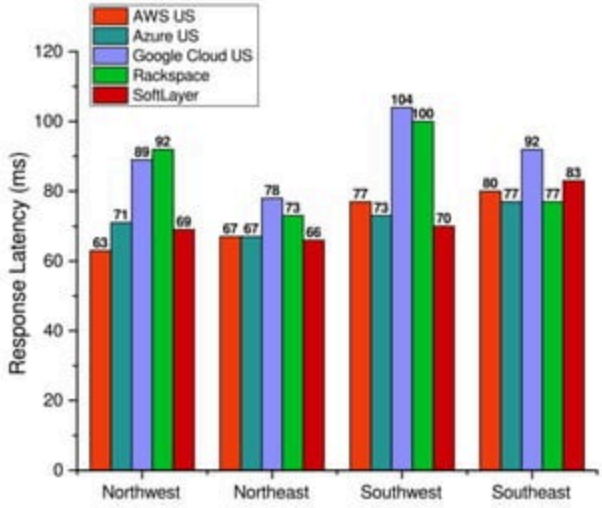
A TIMELINE OF COMPUTATION OFFLOADING - 5



- Cisco announced the role of fog computing in 2012
- ETSI published the technical white paper for MEC in 2014
- 5G drives continuously prosperous development of edge computing: fiber-like high speed and ultra-low latency

THE NEEDS: LATENCY

- Latency crucial for applications like autonomous driving, real-time video analytics and AR/VR
- Average latency RTT in Amazon was 74ms in 2010, and 78ms in 2016



(2016)

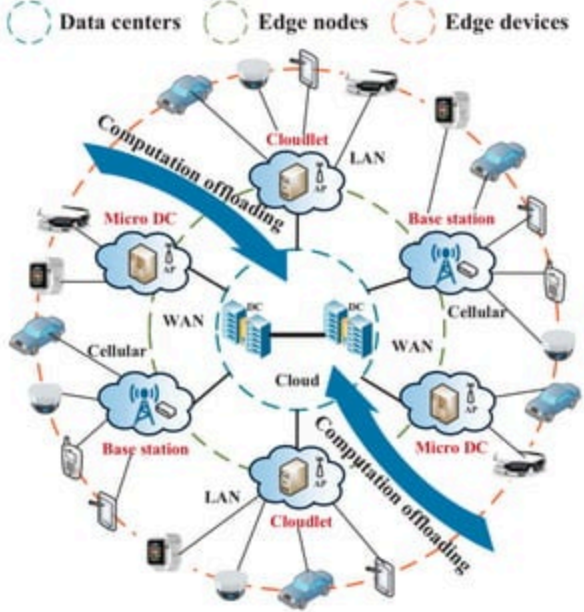
OPPORTUNITIES AND NEEDS

Technology	2.5G(Edge)	3G+(HSPA)	4G(LTE)	5G
Data rate	236 Kbps	22-56 Mbps	100 Mbps	1 Gbps

- 5G has latency of around 1ms
- Computing at the edge can reduce the network traffic of the core network
- "Data deluge": an autonomous vehicle produce 4000-Gb data each day
- Alert systems that integrates AI in the Cloud with AI in the Edge

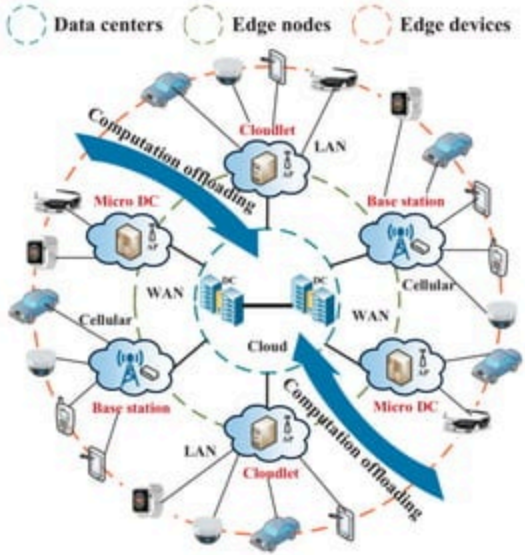
MAP

THE ARCHITECTURE



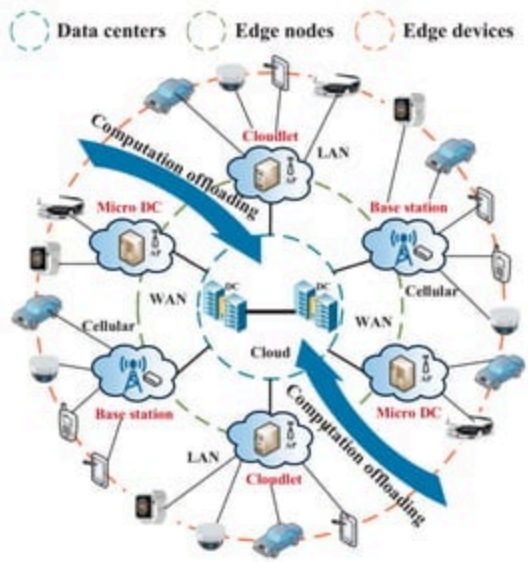
TYPES OF EDGE NODES: CLOUDLETS

- A cloudlet is a resource-rich, trusted, and one-hop network latency to nearby mobile devices
- "A Data Center in a box"
- Deployed at coffee shops, hospitals, airports, stations



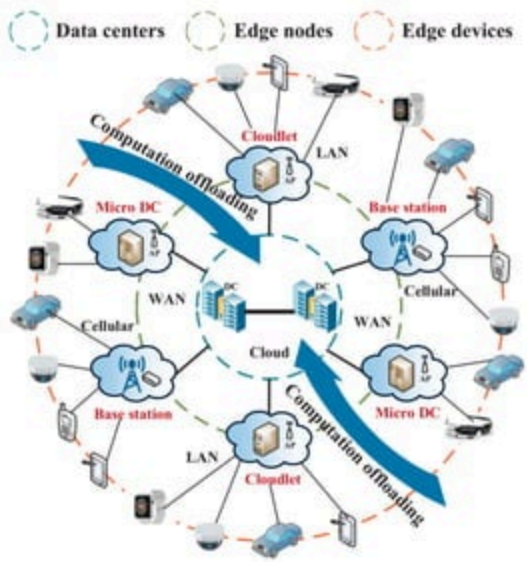
TYPES OF EDGE NODES: MICRO DATA CENTER

- mDCs are geo-distributed small DCs, with small or moderate order of servers.
- Strategically deployed, it can lower the network latency, save bandwidth consumption, provide reliable connectivity, and reduce the overhead of the cloud
- Akamai is a successful example

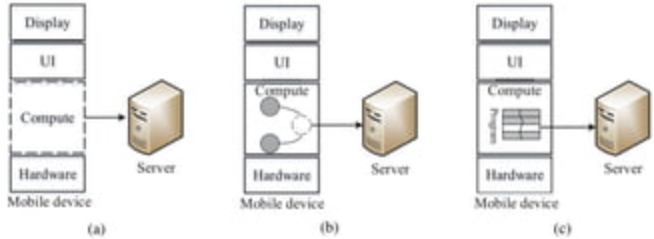


TYPES OF EDGE NODES: BASE STATIONS AND OTHERS

- Communication technologies (such as LTE and 5G) evolved the base stations
- Densely deployed in close proximity to mobile users
- Others: vehicles equipped with onboard computers, IoT gateways, and even smartphones (SociableSense 2011)

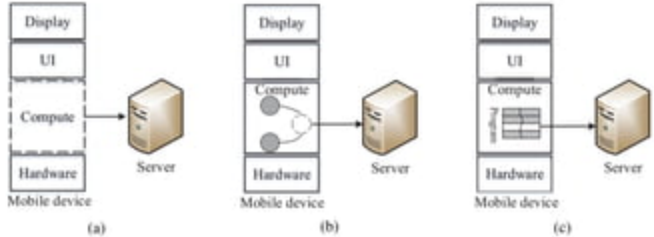


GRANULARITY: FULL



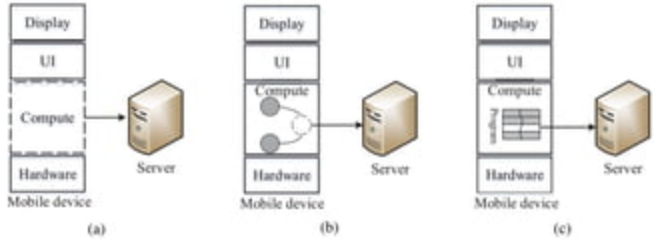
- The whole computing part of applications migrates, leaving mobile devices that are only responsible for UI, input/output, and data sensing.
- Thin clients.
- Web-apps: thin clients only get user input and browse the results.

GRANULARITY: TASK/COMPONENT



- Applications divided and compute-intensive tasks/components to remote infrastructures
- Need to analyze the workflow of the application execution to gain a deep insight into the program behavior.
- Application-dependent
- AR application in 4 steps: video capture, object tracking, scene rendering, and display: offloading object tracking can have a great benefit.

GRANULARITY: METHOD/THREAD



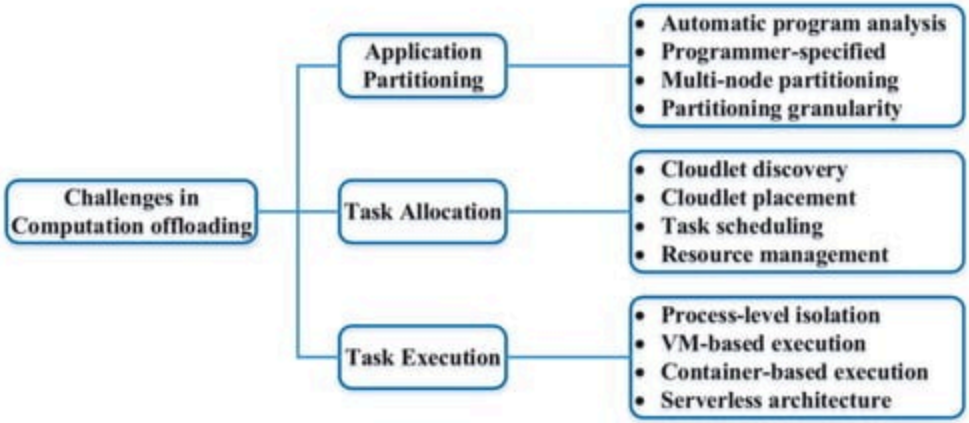
- A fine-grained computing migration
- Needs partitioning mechanisms to assist to find the most beneficial methods.
- Example: function performing optical character recognition (OCR) in an AR application.

IMPACT OF NETWORK CONNECTIVITY

- Both for the latency and the energy consumption.
- Wi-Fi with better performance than with 3G and 4G LTE: reduce latency upto 83% (MAUI)
- In particular, 4G LTE has up to 40% longer network latency than Wi-Fi.
- Wi-Fi is also more energy efficient than 3G and 4G LTE (example: save mode)
- 3G and 4G LTE have high tail energy (stays up with high consumption to anticipate successive transmissions)

MAP

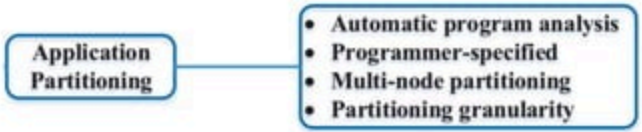
CHALLENGES OF COMPUTATION OFFLOADING



APP PARTITIONING: AUTOMATIC PROGRAM ANALYSIS

- Static or dynamic program analysis techniques to inspect the data flow of application code and analyze potential offloadable parts.
- Graph-based model where vertices indicate methods/objects/tasks and the weights of edges define interaction costs (e.g., data sizes, communication time, and bandwidth).
- Partitioning the graph achieves the desired aim: energy, time, network traffic, ...
- CloneCloud combines static and dynamic profiling: control-flow graph and additional constraints (i.e. methods accessing hw are anchored to devices)

APP PARTITIONING: PROGRAMMER-SPECIFIED

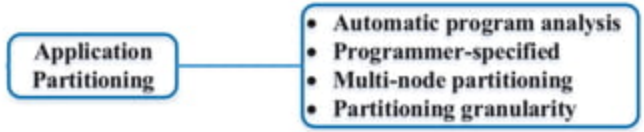


- It offers flexibility (strongly limited by previous approach)
- The hypothesis (MAUI, CloneCloud) "*For every application, the number of useful ways of splitting the application for remote execution is small*"
- MAUI allows programmers to annotate methods considered to be executed remotely with the support of its customized runtime environment.
- Echo adopts Aspect-Oriented Programming to insert additional operations into application code, that manipulate the process of method offloading automatically.

APP PARTITIONING: MULTINODE OFFLOADING

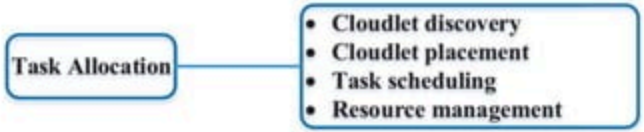
- Specific for Edge
- Tasks in Mobile Cloud Computing have 2 choices to be placed
- In edge computing, 3+ (local, cloud, any of the edge nodes)
- Data-centric applications are suitable for partitioning algorithms (ex. photo recognition and image matching, Flick, Facebook)
- Object interaction graph to represent the program's behavior and a heuristic graph-cutting algorithm to minimize communication costs.

APP PARTITIONING: PARTITION GRANULARITY



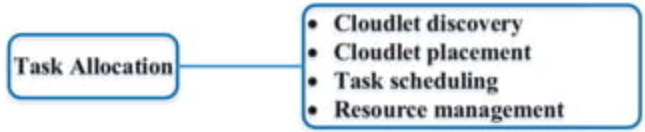
- Offloading toward Cloud works usually also at the method, thread, and class level.
- Not only data dependence but also context dependence: task execution requires consistent runtime environments.
- Edge heterogeneous infrastructures requires complex (expensive) solutions.
- Usually toward Edge, offloading performed at the task/component
- Divide in workflows (tasks/components) with data dependence (pipeline)

TASK ALLOCATION: CLOUDLET DISCOVERY



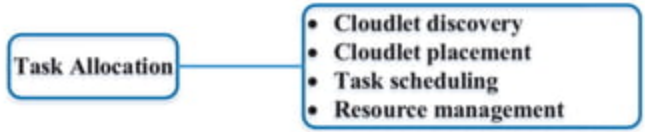
- LAN: zero-conf protocols like Bonjour, Avahi
- WAN: using directory servers
- Application-aware: network-intensive over computation-intensive
- Example: cloud gaming, which is latency-sensitive, the network proximity between mobile devices and cloudlets is the most critical factor in choosing cloudlets.

TASK ALLOCATION: CLOUDLET PLACEMENT



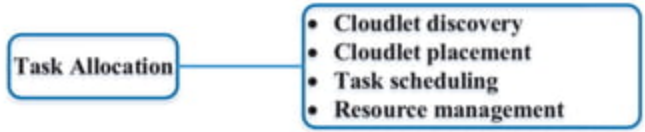
- Placing cloudlets optimally in light of application requirements
- Gedeon et al. proposes a thorough analysis of the existing urban infrastructures, including smart lamp posts, commercial routers, and cellular base stations ...
- ... and a placement strategy to position cloudlets on these existing infrastructures according to the cost and QoS.

TASK ALLOCATION: TASK SCHEDULING



- Task allocation is not only a choice of 2 but on 3+ choices
- Parameters: resources available on the mobile device and resource requirements for task executing
- MAUI combines resource profiling and decision solving.
- Resource profiling: device, program, and network profilers to collect data
- Global optimization solver (integer linear programming) to find the optimal policy that minimizes the energy consumption of mobile devices by satisfying latency constraints.

TASK ALLOCATION: RESOURCE MANAGEMENT

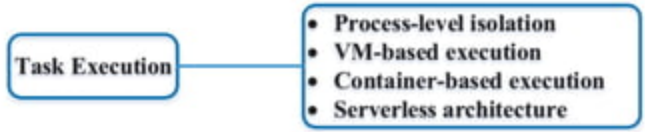


- "Resource" from mobile point of view: energy
- Some produce an energy-efficient resource scheduling policy by adjusting the CPU clock by dynamic voltage and frequency scaling (DVFS).
- ThinkAir realizes an on-demand cloud resource allocation through dynamic VM provisioning (also from cloud perspective with scalability and resources supply).
  - task parallelism based on dynamic scaling of computational power.

DIFFICULTIES OF SCHEDULING

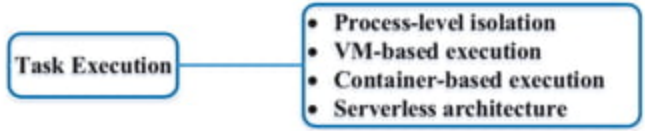
- Twofold challenge
- Single-user scenarios: trading off between the effectiveness and efficiency of resource provisioning.
- Multiuser scenario: multinode collaborated resource allocations and load balancing become important
- Load balancing: cloudlets are geographically distributed without a single centralized node
- Some strategies adopt simple random sampling approach for load balancing based on the theory of “the power of two choices”: scheduler will randomly probe two servers and select the one with the lighter load to place the task.

TASK EXECUTION



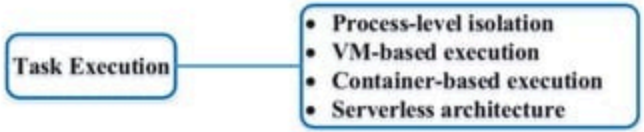
- Task execution spans edge nodes or the cloud: heterogeneous platforms.
- Edge node with the ARM-based chip, while a server runs with x86 CPUs.
- Hardware and software abstractions needed
- Typical solution: start a VM equipped with a mobile OS image
- A wide range of technologies to support this execution, from the early process-level isolation to VM-base solutions to the new serverless architecture.

TASK EXECUTION: PROCESS/VM-LEVEL ISOLATION



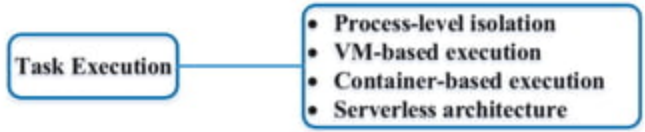
- Initial attempts: Spectra sets up a separate process on the server for handling the code execution via the RPC mechanism.
- Cuckoo uses an interface definition language (AIDL) in Android to describe the code to be offloaded.
- Then, it compiles the code by Java Builder into an application package. then run on a JVM for offloading
- CloneCloud runs tasks based on Android Dalvik VM, and it can automatically offload parts of an application to the clone of a mobile device in the cloud at the thread granularity.
- COMET adopts the basic approaches of CloneCloud and further employs distributed shared memory (DSM) to support multithread tasks.

TASK EXECUTION: CONTAINER-BASED ISOLATION



- Lightweight virtualization solutions (containers and unikernels) well-suited to edge infrastructures
- A container provides isolation on top of the host OS sharing the OS kernel and software stacks: LXC, OpenVZ, and Docker
- Unikernel unique OS kernel containing the minimum OS functions to support task execution (Mirage OS, Exokernel)
- Performances of containers / unikernel outperforms VMs' (disk, memory footprint, and service latency).
- But VMs have advantages on aspects of transparency, isolation, safety, and deployability
- A new approach combining containers / unikernels with VMs: running containers upon VMs

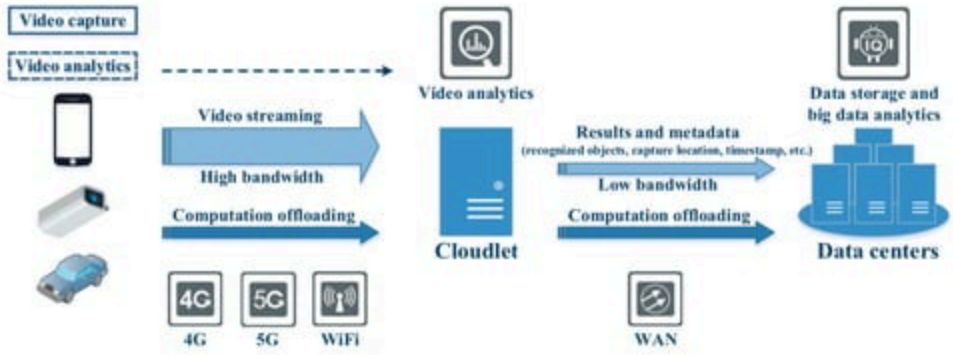
TASK EXECUTION: SERVERLESS ARCHITECTURE



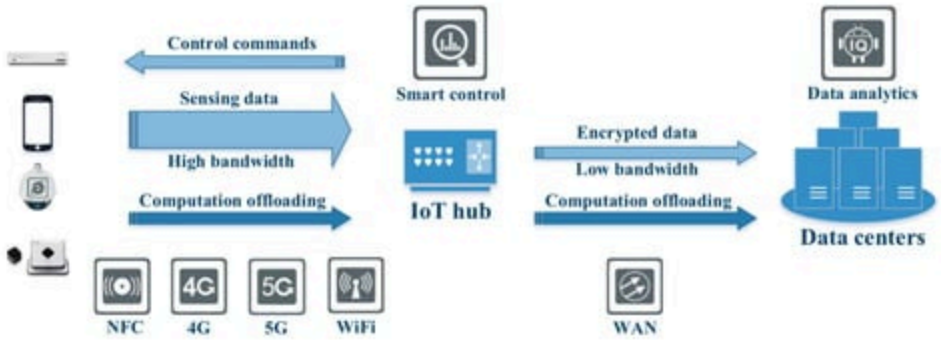
- Serverless computing (FaaS) forces to think of applications as a set of functions.
- No details needed on server deployment, resource allocation, and software configuration (from OSs to runtimes to libraries)
- Good scalability and easy deployment: suitable for edge computing, especially in the area of IoT
- Amazon Lambda, Google Cloud Functions, Microsoft Azure Functions, IBM Cloud Functions, and the open-source OpenLambda

MAP

SCENARIO 1: VIDEO ANALYTICS



SCENARIO 2: SMART HOME/SENSORS



MAP

PROGRAMMING MODEL - 1

- Programming faces many challenges: runtime decision, resource management, and task execution.
- Some early efforts on models like Sapphire: a distributed programming platform
- It separates the application logic from deployment logic, and it provides system components, such as computation offloading, serialization, fault tolerance, and caching.
- Inheritance to equip functions with distributed features
- Twofold challenges: (1) the handling of distributed execution across the three-tier architecture, i.e., mobile devices, cloudlets, and the cloud, and the (2) coordinated resource managing of cloudlets over edge clouds.

PROGRAMMING MODEL - 2

- CloudPath tries to to minimize the complexity of developing and deploying edge applications by providing a RESTful development model to encapsulate computation capabilities based on serverless cloud containers (similar to Amazon AWS Lambda).
- Several commercial unified programming models for cloud/edge
- Azure IoT Edge, Amazon AWS IoT Greengrass, and Cisco IOx [207].
- Common technique: extend cloud capacities to edge infrastructures, enabling the data processing and analysis at the edge of the network and providing unified programming models for application development and deployment.

DOMAIN SPECIFIC LANGUAGES

- A domain-specific programming model for the target application would be more effective than a general one.
- SpanEdge for streaming data (e.g., video streaming), EveryLite for microtasks in IoT, and distributed deep neural network (DDNN)
- SpanEdge: a programming model to specify which portions of an application should be proceeded near the data source.
  - a two-tier data analytics framework: geo-distributed analysis and centralized analysis.
- EveryLite is a scripting language to offer an elastic runtime for microtasks (lightweight tasks running in an embedded OS) in IoT.
- DDNN builds a distributed DNN training model on top of the edge-cloud hybrid architecture.

OFFLOADING AS A SERVICE

- Offloading requests from mobile applications to the cloud
- To extend OaaS to edge computing, needed edge computing infrastructure
- Potential providers are cloud service companies (Amazon, Microsoft, and Google)
- Still lagging behind (in spite of early attempts like AWS IoT Greengrass)
- Decentralized solutions including volunteering computing with any possible users' machines, including PCs, private servers, and entire Data Centers
- Named Function as a Service enables the dynamic execution of user code without knowing edge node provi- sioning, by using serverless architecture and building on unikernels.

READING MATERIAL AND CREDITS

- L. Lin, X. Liao, H. Jin and P. Li, "Computation Offloading Toward Edge Computing," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1584-1607, Aug. 2019, doi: 10.1109/JPROC.2019.2922285.

QUESTIONS? COMMENTS? ANYTHING?


That's all Folks!