VERS UNE APPROCHE FAAS
POUR DU CALCUL ANALYTIQUE
SUR NOEUDS IOT

DAVID FERNÁNDEZ BLANCO & FRÉDÉRIC LE MOUËL
Agenda

1. Context - Issues
2. Architecture
3. Results
4. Next steps
Issues

- The applications demand lots of resources (computation, storage) & functionalities (availability, dependability) that's why we execute them in the cloud.
- The geographical distance to the remote clouds and sometimes the poor networking limit the applications, worsening the response delay & the QoS of our applications.
### Current works

<table>
<thead>
<tr>
<th>Mobile CC</th>
<th>Classic CC</th>
<th>QOS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resources Management</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Energy Management</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Mobility Management</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Task Scheduling</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Offloading</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Fault Tolerance</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Data Extraction</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Load Balancing</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Graph Analytics</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Bandwidth Allocation</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Reduce Latency</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Flexibility</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

---

*List of works:*
- Precise Edge Computing with Hard Deadlines
- Exploiting smart e-health gateways at the edge of health care internet of things
- An Energy-Aware IoT Fog/cloud System
- FCIAT: Integrated Fog Cloud IoT Architectural Paradigm for Future Internet of Things
- A Cloudlet-based Mobile Computing Model for Resource and Energy Efficient Offloading
- CloudFAtt: A Multi-Tier Cloud Computing Framework
- PyCar
- Serverless Computation with...

---

*EMCloud: A Resilient, secure, cloud with exploitable mobile devices*

*ruCloud: Volunteer Computing as a Service (Cloud) System*
## Most fitted approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>Resource Management</th>
<th>Energy Consumption</th>
<th>Mobility Management</th>
<th>Task Scheduling</th>
<th>Offloading</th>
<th>Fault Tolerance</th>
<th>Data replication</th>
<th>Load Balancing</th>
<th>Graph partitioning</th>
<th>Scalability</th>
<th>Reliability</th>
<th>Bandwidth optimisation</th>
<th>Reduce Latency</th>
<th>Flexibility</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMC: A hierarchical volunteer cloud with explicit mobile devices</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>cuCloud: Volunteer Computing as a Service (VCaas) System</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
CONCLUSION

- Conclusions: There are no approaches aiming at offloading to close nodes - to reduce latency - being really constrained - such as IoT nodes.
Faas - Serverless Cloud Computing [2]

- Serverless doesn't mean "no server".
- An event may trigger the execution of a function or a number of functions in parallel.
- AWS Lambda, IBMOpenWhisk and Google Cloud Functions
Architecture
PyCloudIoT

Cluster 1
ESP-32
Store libraries, Run functions, Cache, Consensus

Cluster 2
ESP-32
Store libraries, Run functions, Cache, Consensus

Serveur Mosquitta
Ip, réseau, port, topics...

Dispatcher
Send/receive messages
Cluster management
Cache functions

Dispatcher: HP Elite Book

Client.py
Compile the program, send requests, receive results...
Dynamic Clustering

- The nodes send Network discovery messages and a cluster is assigned to them. In this case, via a message, the dispatcher transmits the channels from this cluster:
  - Leader Channel
  - Broadcast Channel
  - Final answer channel
Consensus

- Basic democracy consensus in which in case of tie, the answer will be the one said by the leader or (if the leader has an answer with a lower number of votes than the ones tied) the first one that arrived to the leader.

- The leader is the most ancient node in the cluster and will wait twice the sleep period of the slowest node to decide which is the answer.
Cache and Sleep strategy

- Cache strategy remembering the last 8 function results. Round Robin.

- Sleep strategy, nowadays the service does not stop, there's always an active callback, it will be changed by periodically check if there are messages for his topic on the MQTT server.
Parsing

For the libraries:
- #PYCLOUDIOT : LIBRARY, beginning line, ending line, class name,
  #IMPORTS : library 1 ; library 2 ....

For the scripts to execute:
- #PYCLOUDIOT : MAIN, beginning line, ending line, class name,
  #IMPORTS : library 1 ; library 2 ....
 Parsing

```python
#PYCLOUDIOT : LIBRARY,2,10,fibonacci_library.py,
def Fibonacci(n=40):
    if n<0:
        print("Incorrect input")
    elif n==1:
        return 0
    elif n==2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)

#PYCLOUDIOT : MAIN,13,17,fibonacci_main.py, #IMPORTS :fibonacci_library.py;sys
a = 1
b = 2
c = 3
to_calculate = a + b + c
print(Fibonacci(to_calculate))
```

Library
```
def Fibonacci(n=40):
    if n<0:
        print("Incorrect input")
    elif n==1:
        return 0
    elif n==2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)
```

Main
```
from fibonacci_library import *
import sys

a = 1
b = 2
c = 3
to_calculate = a + b + c
print(Fibonacci(to_calculate))
```
First results

Tests done with 5 clusters for the calculation of the fibonacci number on either a macbook pro 15 (in blue) and the infrastructure PyCloudIOT (in orange).
First results

We have a key to decrypt and get the $p$ and $q$ factors of the number. This test has been done with a 5 cluster architecture, a dispatcher server and a single client. We have used the pollard-rho algorithm.
Next Steps?

- Intelligent parsing based on the application analysis and energy consumption models.
- Reinforce the fault tolerance by evolving the consensus algorithm.
- Performance improvements to prevent memory faults.
- Defining a cache policy based on the popularity of the scripts.
- Testing with a macro-benchmark.
References


QUESTIONS