Leveraging Constrained Devices for Custom Code Execution in the Internet of Things

Flavia Pisani (fpisani@ic.unicamp.br), Advisor: Professor Edson Borin, Ph.D.

Institute of Computing — University of Campinas

Acknowledgments:

CAPES, CNPq, FAPESP, LMCAD, Motorola

Internet of Things (IoT)

- Tens of billions of devices connected to the IoT in the next decade
  - Petabytes/day scale
  - Moving data from sensors to servers is expensive in many ways: **Time** **Money** **Energy**
- Possible solution: bring the computation closer to the data
- Users may want to query IoT devices to analyze their data
  - Requires custom code execution support

Fog Computing

- Fog can be defined as a cloud that is “close to the ground”
- Takes the computation closer to the data
- Runs typically (not exclusively) on network edge devices
- New possibilities: **Finding outliers** **Filtering data** **Meeting real-time requirements**

Constrained Devices

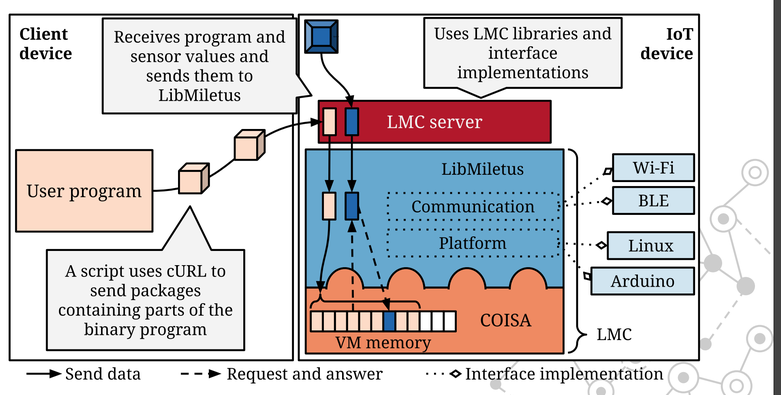
- In this scenario, many devices will be resource-constrained
  - Current fog frameworks are too large for them
- The Internet Engineering Task Force definition of “constrained devices”:
  - Tight limits on power, memory, and processing resources

Name	Data size (e.g., RAM)	Code size (e.g., Flash)
Class 0 (C0)	≪ 10 KiB	≪ 100 KiB
Class 1 (C1)	~ 10 KiB	~ 100 KiB
Class 2 (C2)	~ 50 KiB	~ 250 KiB
Others	≫ 50 KiB	≫ 250 KiB

- These boundaries will change over time, but Moore’s law tends to be less accurate for embedded devices

Custom Code Execution on Constrained Devices

- **Framework: LibMiletusCOISA (LMC) = LibMiletus (library) + COISA (VM)**
  - LibMiletus:
    - Enables developers to easily design and implement IoT devices
    - Hides communication specifics
      - Focuses on device functionalities instead of system infrastructure
    - Compatible with low-cost devices
    - Easily installed and used with Arduino IDE
      - Accessible to programmers with little or no experience with embedded systems
  - COISA:
    - Compact VM: runs on platforms with 2 kB of RAM
    - Uses OpenISA 0.1 (this ISA has evolved since then)
      - Compatible with MIPS
      - Emulates guest applications on ARM and x86 host processors at near native performance



Publication

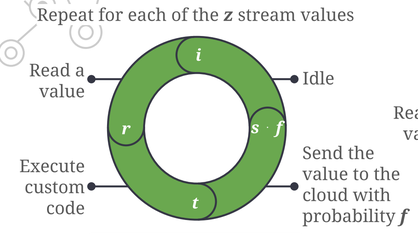
F. Pisani, J. R. Brunetta, V. M. do Rosario, and E. Borin. **Beyond the Fog: Bringing Cross-Platform Code Execution to Constrained IoT Devices**. In Proc. SBAC-PAD ’17, pages 17–24, October 2017. doi: 10.1109/SBAC-PAD.2017.10.

Modeling Cloud and Fog Execution

- We are particularly interested in modeling the execution of filtering procedures
  - Can be simple enough to run on constrained devices
  - Have the potential to drastically decrease the amount of data sent to the cloud
  - Can be modeled by the probability that a value will pass the filter (which we call f)
- **Mathematical model:**

Fog computing cost (C_F)

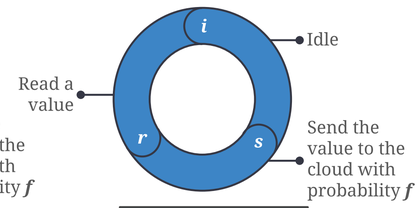
Repeat for each of the z stream values



$$C_F = (r + t + s \cdot f + i) \cdot z$$

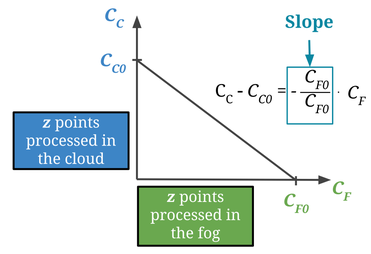
Cloud computing cost (C_C)

Repeat for each of the z stream values

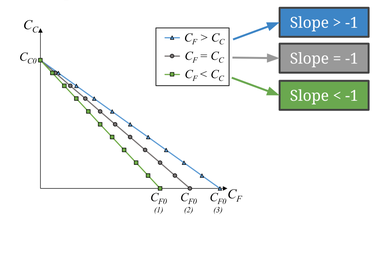


$$C_C = (r + s + i) \cdot z$$

- **Visual model:**



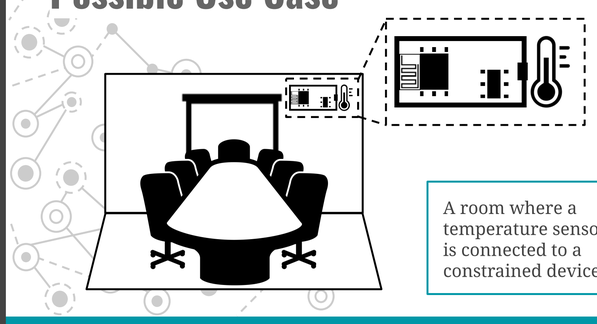
- Exemple for z = 10:



Publication

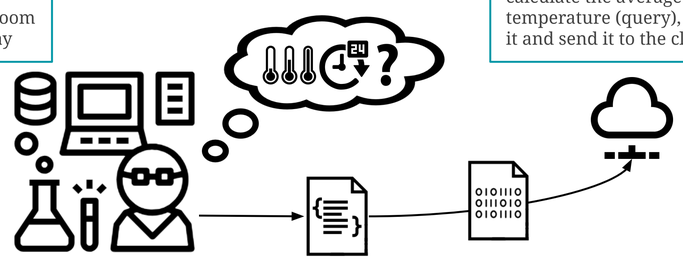
F. Pisani, V. M. do Rosario, E. Borin. **Fog vs. Cloud Computing: Should I Stay or Should I Go?**. Future Internet, 11(2):27–32, February 2019. doi: 10.3390/fi11020034.

Possible Use Case



A room where a temperature sensor is connected to a constrained device

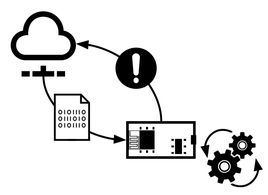
A data scientist wants to find out the average temperature of the room in the span of one day



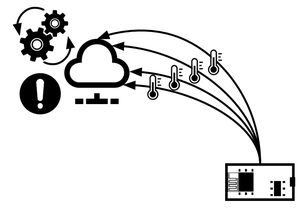
They write a program to calculate the average temperature (query), compile it and send it to the cloud

Where should we run the query?

Fog Computing



Cloud Computing



Devices

- **NodeMCU 1.0 (ESP8266, ESP-12E)**
  - RAM: 64 KiB (instruction) + 96 KiB (data)
  - Flash: 1 MiB + 3 MiB (File System)
  - CPU: Tensilica Xtensa L106 @ 80 MHz

- **DragonBoard 410c**
  - RAM: 1 GB LPDDR
  - Flash: 8 GB
  - CPU: Quad-core ARM Cortex A53 Snapdragon 410E up to 1.2 GHz

Apache Edgent

- We compare **LMC** to **Edgent**
  - Apache Incubator programming model and runtime
  - Both are open source and perform the computation on devices where the data is being collected
  - Written in Java, so needing the JVM makes it incompatible with NodeMCU

Benchmarks

- **MinMax**
  - Simple filter: value < min OR value > max
  - Detects malfunctions in systems with known lower and upper bounds
  - “MinMax [-5, 5]” filters out numbers in the [−5, 5] range
- **Outlier**
  - Tukey’s test on a window of size N
  - Detects possible anomalies or failures in data models
  - “Outlier 16” finds outliers in a window of 16 values
- **FFT**
  - Fast Fourier Transform on a window of size N (recursive)
  - Useful as a step in voice activity detection

Datasets

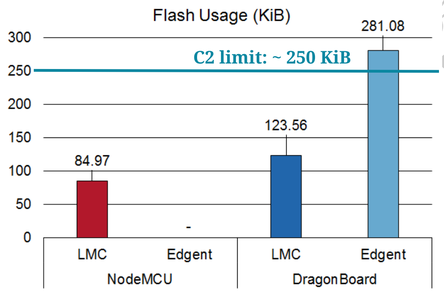
- Four real datasets (subset of the hourly local climatological data collected at Chicago O’Hare International Airport between September 2008 and August 2018):
  - **HRelHumidity**: relative humidity given to the nearest whole percentage (16 to 100)
  - **HVisibility**: horizontal distance an object can be seen and identified given in whole miles (0 to 10)
  - **HWBTempC**: wet-bulb temperature given in tenths of a degree Celsius (−27.3 to 27.4)
  - **HWindSpeed**: speed of the wind at the time of observation given in miles per hour (0 to 55)
- One artificial dataset:
  - **Synthetic**: random floating-point numbers generated using the sine function combined with a Gaussian error (−84.05 to 85.07)

Additional Information

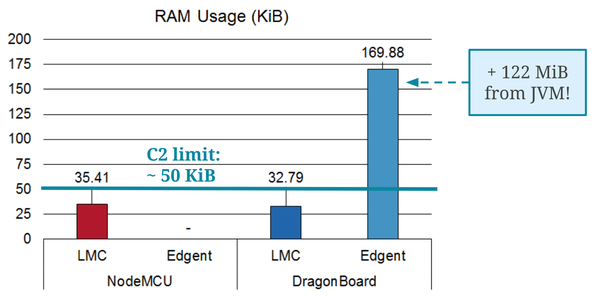
- **Experiments - Custom Code Execution on Constrained Devices:**
  - We compare the LMC COISA VM running with interpretation with Edgent running on the JVM with only interpretation by turning off JVM’s just-in-time (JIT) mechanism
  - The COISA VM can employ Cloud-Assisted Translation to accelerate the code emulation similarly to JIT, as this mechanism is not currently implemented, we estimate the performance by compiling and running the user code as native code
  - Synthetic dataset
- **Experiments - Modeling Cloud and Fog Execution:**
  - In the mathematical model, the user can define a cost type (e.g., number of instructions, execution time, energy consumption) and plug in values to analyze test cases
  - Given that f depends on the data, we can estimate it based on a subset of the stream (“Est.” results) and compare it to the f value considering the whole stream (“Real” results)
  - Fog-prone cases: the fog is more likely to be profitable, as s is about one order of magnitude larger than t
  - Cloud-prone cases: the cloud is more likely to be profitable when s and t are close
  - We can vary s and t and see how that would impact our decision
    - The dashed area in the simulation tables represents the values used in the graphs
    - The cells below the continuous line are the ones where the slope is ≤ -1(fog computing is more profitable), the others are the cases where cloud computing has the lower cost
  - z = 65,536

Custom Code Execution Experiments

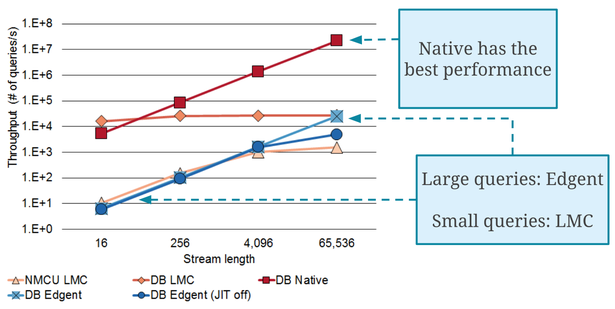
- Code Size



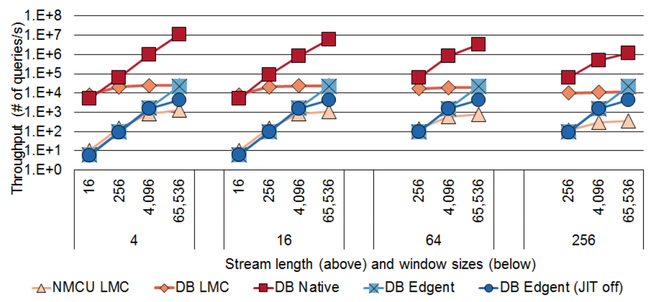
- Data Size



- Events Processed per Second (MinMax [-15, 15])



- Events Processed per Second (Outlier, FFT is similar)

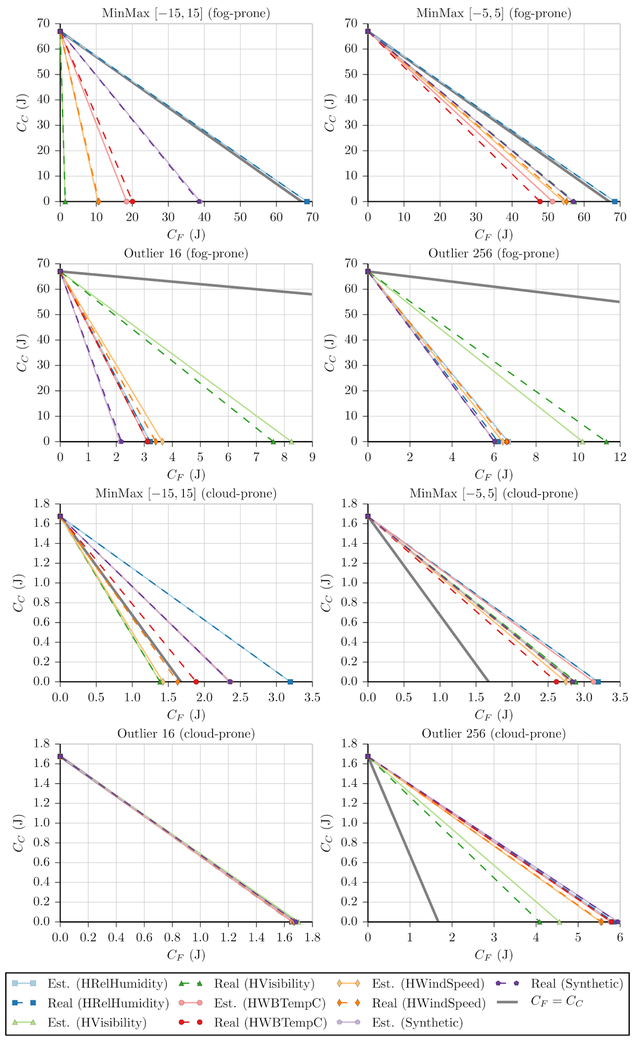


We showed that LMC is:

- Very compact and compatible with Class 2 constrained devices
- Static translation: overall faster than Edgent
- Interpretation: faster than Edgent at lightweight quick queries (in some cases, even for LMC+NodeMCU vs. Edgent+DragonBoard)

Modeling Experiments

- Graphs of the linear equations for each dataset and benchmark using energy consumption as the cost (NodeMCU vs. Cloud):



- Slope simulations for each test case using energy consumption as the cost (NodeMCU vs. Cloud):

s (mJ) \ t (mJ)	0.02	0.07	0.12	0.17	0.22	0.27	0.32	0.37	0.42	0.47
0.01	-0.39	-0.13	-0.08	-0.06	-0.04	-0.04	-0.03	-0.03	-0.02	-0.02
0.16	-1.46	-1.00	-0.76	-0.62	-0.52	-0.45	-0.39	-0.35	-0.31	-0.29
0.31	-1.60	-1.27	-1.06	-0.90	-0.79	-0.70	-0.63	-0.57	-0.52	-0.48
0.46	-1.66	-1.41	-1.22	-1.08	-0.96	-0.87	-0.80	-0.73	-0.68	-0.63
0.61	-1.69	-1.48	-1.32	-1.19	-1.09	-1.00	-0.92	-0.86	-0.80	-0.75
0.76	-1.71	-1.54	-1.39	-1.28	-1.18	-1.09	-1.02	-0.96	-0.90	-0.85
0.91	-1.72	-1.57	-1.45	-1.34	-1.25	-1.17	-1.10	-1.04	-0.98	-0.93
1.06	-1.73	-1.60	-1.49	-1.39	-1.30	-1.23	-1.16	-1.10	-1.05	-1.00
1.21	-1.74	-1.62	-1.52	-1.43	-1.35	-1.28	-1.21	-1.16	-1.10	-1.06
1.36	-1.74	-1.64	-1.54	-1.46	-1.39	-1.32	-1.26	-1.20	-1.15	-1.11

(a) MinMax [−15, 15].

s (mJ) \ t (mJ)	0.02	0.04	0.06	0.08	0.10	0.12	0.14	0.16	0.18	0.20
0.01	-0.35	-0.21	-0.15	-0.11	-0.09	-0.08	-0.07	-0.06	-0.05	-0.05
0.16	-1.05	-0.93	-0.83	-0.75	-0.69	-0.63	-0.59	-0.55	-0.51	-0.48
0.31	-1.12	-1.04	-0.98	-0.92	-0.87	-0.82	-0.78	-0.74	-0.71	-0.68
0.46	-1.15	-1.09	-1.04	-1.00	-0.96	-0.92	-0.88	-0.85	-0.82	-0.79
0.61	-1.16	-1.12	-1.08	-1.04	-1.01	-0.97	-0.94	-0.92	-0.89	-0.86
0.76	-1.17	-1.13	-1.10	-1.07	-1.04	-1.01	-0.99	-0.96	-0.94	-0.92
0.91	-1.18	-1.15	-1.12	-1.09	-1.07	-1.04	-1.02	-1.00	-0.97	-0.95
1.06	-1.18	-1.15	-1.13	-1.11	-1.08	-1.06	-1.04	-1.02	-1.00	-0.98
1.21	-1.18	-1.16	-1.14	-1.12	-1.10	-1.08	-1.06	-1.04	-1.02	-1.01
1.36	-1.19	-1.16	-1.15	-1.13	-1.11	-1.09	-1.07	-1.06	-1.04	-1.02

(b) MinMax [−5, 5].

s (mJ) \ t (mJ)	0.02	0.12	0.22	0.32	0.42	0.52	0.62	0.72	0.82	0.92
0.01	-0.50	-0.08	-0.05	-0.03	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01
0.16	-7.59	-1.32	-0.72	-0.50	-0.38	-0.31	-0.26	-0.22	-0.19	-0.17
0.31	-14.01	-2.54	-1.40	-0.96	-0.73	-0.59	-0.50	-0.43	-0.38	-0.34
0.46	-19.87	-3.74	-2.06	-1.42	-1.09	-0.88	-0.74	-0.64	-0.56	-0.50
0.61	-25.24	-4.91	-2.72	-1.88	-1.44	-1.16	-0.98	-0.84	-0.74	-0.66
0.76	-30.16	-6.07	-3.37	-2.34	-1.79	-1.45	-1.22	-1.05	-0.92	-0.82
0.91	-34.70	-7.21	-4.02	-2.79	-2.14	-1.73	-1.45	-1.25	-1.10	-0.98
1.06	-38.90	-8.33	-4.66	-3.24	-2.48	-2.01	-1.69	-1.46	-1.28	-1.14
1.21	-42.80	-9.43	-5.30	-3.69	-2.83	-2.29	-1.93	-1.66	-1.46	-1.30
1.36	-46.42	-10.52	-5.93	-4.13	-3.17	-2.57	-2.16	-1.86	-1.64	-1.46

(c) Outlier 16.

s (mJ) \ t (mJ)	0.02	0.14	0.26	0.38	0.50	0.62	0.74	0.86	0.98	1.10
0.01	-0.50	-0.07	-0.04	-0.03	-0.02	-0.02	-0.01	-0.01	-0.01	-0.01
0.16	-7.91	-1.14	-0.61	-0.42	-0.32	-0.26	-0.22	-0.19	-0.16	-0.15
0.31	-15.15	-2.21	-1.19	-0.81	-0.62	-0.50	-0.42	-0.36	-0.32	-0.28
0.46	-22.23	-3.27	-1.76	-1.21	-0.92	-0.74	-0.62	-0.53	-0.47	-0.42
0.61	-29.17	-4.33	-2.34	-1.60	-1.22	-0.98	-0.82	-0.71	-0.62	-0.55
0.76	-35.95	-5.38	-2.91	-1.99	-1.52	-1.22	-1.03	-0.88	-0.77	-0.69
0.91	-42.60	-6.44	-3.48	-2.39	-1.82	-1.46	-1.23	-1.06	-0.93	-0.83
1.06	-49.10	-7.49	-4.05	-2.78	-2.11	-1.71	-1.43	-1.23	-1.08	-0.96
1.21	-55.47	-8.53	-4.62	-3.17	-2.41	-1.95	-1.63	-1.40	-1.23	-1.10
1.36	-61.72	-9.58	-5.19	-3.56	-2.71	-2.19	-1.83	-1.58	-1.38	-1.23

(d) Outlier 256.

- Mathematical + visual models
  - Help the user to decide which of the platforms presents the lowest cost according to a certain metric
  - Linear equations, decide by analyzing the slope of a line
    - Slope ≤ -1 → fog, cloud otherwise
- The model only chose the incorrect approach for the cloud-prone Outlier 16 test case, where all lines are very close to $C_F = C_C$
- Estimated and real values lead to the same decisions