Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Trabajo de Fin de Máster – Máster Universitario en Inteligencia Computacional e Internet de las Cosas

Universidad de Córdoba
2022/2023

Trabajo realizado por:
-Antonio Gómez Giménez (i72gogia@uco.es)

Directores:
-Fernando León García
-José Manuel Palomares Muñoz

Agradecimientos

- A **Fernando León García,** Profesor Ayudante Doctor, del Departamento de Ingeniería Electrónica y de Computadores de la Universidad de Córdoba.

- A **José Manuel Palomares Muñoz**, Profesor Titular de Universidad, del Departamento de Ingeniería Electrónica y de Computadores de la Universidad de Córdoba.

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y objetivos a cumplir.
- Métodos y materiales.
- Metodología:
- Resultados y discusión.
- Conclusión.

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- **¿Por qué se escoge este proyecto?.**
- Hipótesis y objetivos a cumplir.
- Métodos y materiales.
- Metodología:
- Resultados y discusión.
- Conclusión.

¿Por qué se escoge este proyecto?

- **Problemas existentes con el incremento de uso de IoT** y los grandes **flujos de datos**.
- Problemas de procesamiento, **¿donde se realiza?**
- **No existe metodología única y definida** frente a estos problemas.
- **Proponer una solución** frente a este problema actual, **aplicando stream processing y el paradigma de publicador suscriptor**.

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- **Hipótesis y Objetivos a cumplir.**
- Métodos y materiales.
- Metodología:
- Resultados y discusión.
- Conclusión.

Hipótesis y Objetivos a cumplir.

Hipótesis:

"Creemos que el **paradigma publicador-suscriptor** es idóneo como base tecnológica para desarrollar un **sistema distribuido reconfigurable y escalable**, como es el caso de los **Stream Processing Engines**. Esto se debe fundamentalmente a la facilidad que introduce este paradigma para **crear taxonomías descriptivas**; los mecanismos que aporta para **distribuir la información entre los nodos de la red de manera selectiva**; y la **simplicidad de los mecanismos de suscripción, de-suscripción y publicación**."

Hipótesis y Objetivos a cumplir.

Objetivos:

- **OB-01-> Desarrollar una metodología para Stream Processing reconfigurable** en entornos locales distribuidos con dispositivos de baja capacidad de cómputo.
- **OB-02-> Determinar la viabilidad** de utilizar tecnologías de comunicación local basadas en el **paradigma publicador-suscriptor** para dar soporte al control de un **Stream Processing Engine**.
- **OB-03-> Evaluar el rendimiento** del desarrollo propuesto en un caso de estudio real.

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- **Métodos y materiales.**
  - Paradigma publicador-suscriptor
  - Plataformas hardware en Edge
  - Stream Processing Engines orientados a Edge
  - Network Pipelining
  - Otros conceptos
- Metodología:
- Resultados y discusión.
- Conclusión.

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- **Métodos y materiales.**
  - **Paradigma publicador-suscriptor**
  - Plataformas hardware en Edge
  - Stream Processing Engines orientados a Edge
  - Network Pipelining
  - Otros conceptos
- Metodología:
- Resultados y discusión.
- Conclusión.

Paradigma publicador-suscriptor

- Es un **método de comunicación**.
- Se centra en la **comunicación en grupo**.
- **Divide la información** de manera más **cómoda y eficiente**.
- **No como comunicación multicast**, los clientes del modelo publicador-suscriptor describen los eventos en los que están interesados de una forma más específica (topics).
- Los clientes pueden **publicar notificaciones o suscribirse a los filtros** (topics por ejemplo)
- **Brokers y clientes**.
- **Permite escalabilidad y distribución de la red** permitiendo gran número de clientes y comunicaciones entre ellos.

Paradigma publicador-suscriptor

- Existencia de **distintos protocolos que se basan en el paradigma publicador-suscriptor** para su funcionamiento.
- En nuestro caso se usa **MQTT** (Message Queue Telemetry Transport).
  - **Ligero y eficiente.**
  - **Escalable.**
  - **Fiable.**
  - **Seguro.**
  - **Admitido**

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- **Métodos y materiales.**
  - Paradigma publicador-suscriptor
  - **Plataformas hardware en Edge**
  - Stream Processing Engines orientados a Edge
  - Network Pipelining
  - Otros conceptos
- Metodología:
- Resultados y discusión.
- Conclusión.

Plataformas hardware en Edge

- **¿La computación en el Edge?**
  - Computación cerca de la fuente de datos o muy cerca a la misma.
  - En auge debido a IoT.
  - Para lograr dichos objetivos, existen módulos y componentes, como las **Placas Espressif**:
    - **ESP32** (en nuestro caso)
    - **ESP8266**
    - **ESP32-S**

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- **Métodos y materiales.**
  - Paradigma publicador-suscriptor
  - Plataformas hardware en Edge
  - **Stream Processing Engines orientados a Edge**
  - Network Pipelining
  - Otros conceptos
- Metodología:
- Resultados y discusión.
- Conclusión.

Stream Processing Engines orientados a Edge

- **¿Qué es Stream Processing?**
  - Surge de la idea de procesar los datos de forma continua.
  - Los datos se procesan de forma continua.
  - Para lograrlo, flujos de datos infinitos y sin límites de tiempo.
- La unión de ambos conceptos genera el concepto de **Stream Processing Engines pero con la orientación a Edge.**
- **Procesamiento del flujo de datos pero orientado a los dispositivos del Edge.**
- Existen herramientas basadas en este concepto como **kafka** o **spark.**

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- **Métodos y materiales.**
  - Paradigma publicador-suscriptor
  - Plataformas hardware en Edge
  - Stream Processing Engines orientados a Edge
  - **Network Pipelining**
  - Otros conceptos
- Metodología:
- Resultados y discusión.
- Conclusión.

Network Pipelining

- **¿Qué es Network Pipelining?**
  - Viene de la agrupación de distintos **pipelines**, siendo un **pipeline** el procesamiento que se le aplica a un flujo de datos.
  - Dicho pipeline contiene **diferentes stages** (etapas de procesamiento).
- **Network Pipelining** permite la **paralelización del procesamiento**, existiendo distintos pipelines que realizan el procesamiento sobre etapas.

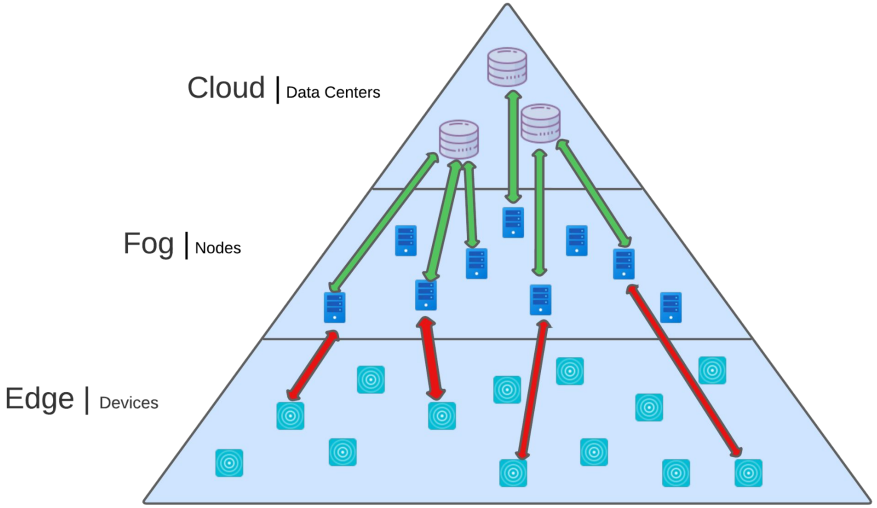
Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- **Métodos y materiales.**
  - Paradigma publicador-suscriptor
  - Plataformas hardware en Edge
  - Stream Processing Engines orientados a Edge
  - Network Pipelining
  - **Otros conceptos**
- Metodología:
- Resultados y discusión.
- Conclusión.

Otros conceptos

- **Estructura de la red:**



Otros conceptos

- Lenguajes de programación -> En nuestro caso **python** (paho-mqtt o umqttsimple)
- Plataformas de integración de IoT en la nube -> **Ubidots**

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- Métodos y materiales.
- **Metodología:**
  - Diseño
  - Implementación
- Resultados y discusión.
- Conclusión.

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- Métodos y materiales.
- **Metodología:**
  - **Diseño**
  - Implementación
- Resultados y discusión.
- Conclusión.

Diseño

- Diseño general del sistema.
- Explicación casos de uso.
- Explicación árbol de topics.

Diseño

- **Diseño general del sistema.**
- Explicación casos de uso.
- Explicación árbol de topics.

Diseño

- **Diseño general del sistema.**

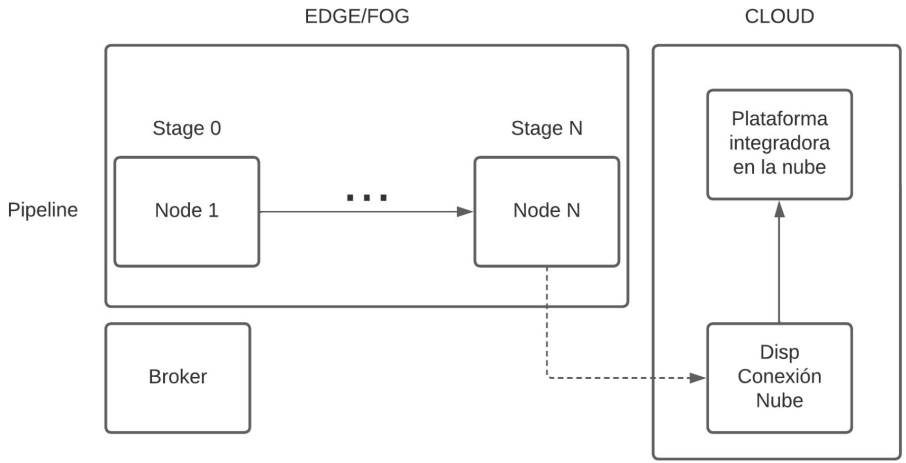


Diagrama del problema en una versión sencilla

Diseño

- **Diseño general del sistema.**

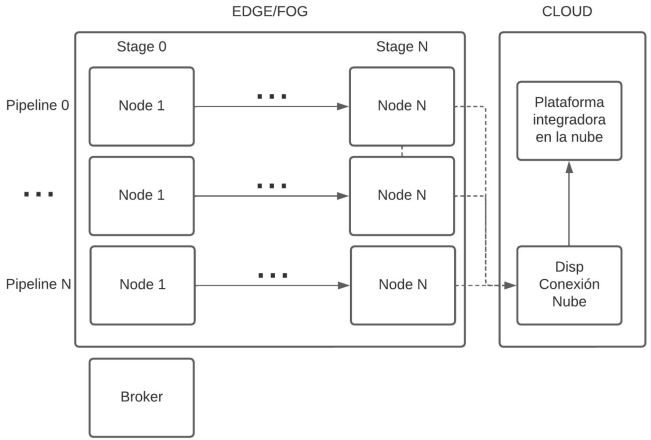


Diagrama del problema en una versión compleja (MQTT)

Diseño

- **Diseño general del sistema.**

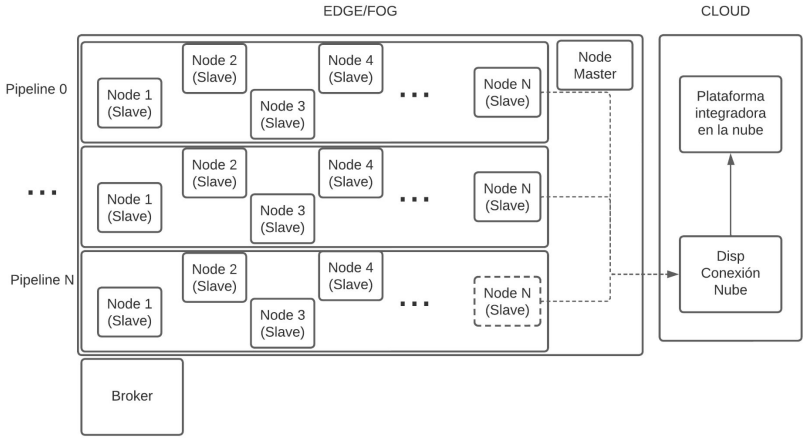


Diagrama que representa la solución que se pretende dar por la librería

Diseño

- Diseño general del sistema.
- **Explicación casos de uso.**
- Explicación árbol de topics.

Diseño

- **Explicación casos de uso.**

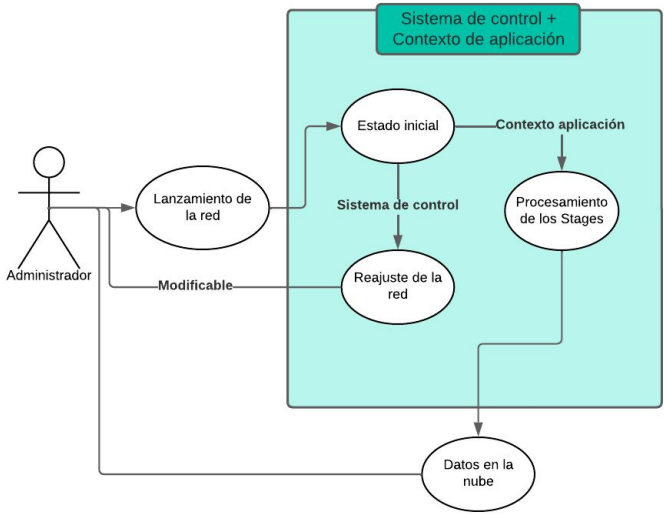


Diagrama del funcionamiento a grandes rasgos de todo el sistema

Diseño

- Diseño general del sistema.
- Explicación casos de uso.
- **Explicación árbol de topics.**

Diseño

- **Explicación árbol de topics.**
  - **Topics contexto de aplicación:**
    - /APPLICATION_CONTEXT/ID-X
    - /APPLICATION_CONTEXT/PIPELINE-W/RESULT

Diseño

- **Explicación árbol de topics.**
  - **Topics de control:**
    - **Master:**
      - /CONTROL/MASTER/GET_MY_ID
      - /CONTROL/MASTER/ID-X/GET_CONNECT_NODES
      - /CONTROL/MASTER/ID-X/GIVE_INFO
    - **Slaves:**
      - /CONTROL/SLAVE/SET_MY_ID/X
      - /CONTROL/SLAVE/ID-X/NEW_SUSCRIBER
      - /CONTROL/SLAVE/ID-X/NEW_PUBLISHER
      - /CONTROL/SLAVE/REQUEST_INFO
      - /CONTROL/SLAVE/ID-X/UPDATE_STAGE

Diseño

- **Explicación árbol de topics.**
  - **Topics conexión a la nube (ubidots):**
    - /v1.6/devices/prueba-stream-processing

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- Métodos y materiales.
- **Metodología:**
  - Diseño
  - **Implementación**
- Resultados y discusión.
- Conclusión.

Implementación

- Servidor MQTT.
- Librería STRPLibrary.
- Script python MasterControl.
- Script python SlaveXControl.
- Script NodeConexCloud.
- Script auxiliar.

Implementación

- **Servidor MQTT.**
- Librería STRPLibrary.
- Script python MasterControl.
- Script python SlaveXControl.
- Script NodeConexCloud.
- Script auxiliar.

Implementación

- Servidor MQTT.
- **Librería STRPLibrary.**
  - Clase Master.
  - Clase Slave.
- Script python MasterControl.
- Script python SlaveXControl.
- Script NodeConexCloud.
- Script auxiliar.

Implementación

- Clase Master

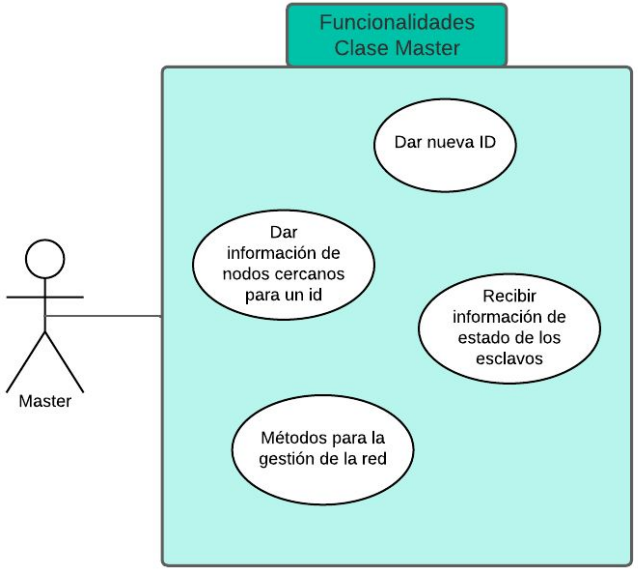


Diagrama de las funcionalidades de la clase master

Implementación

- Clase Slave

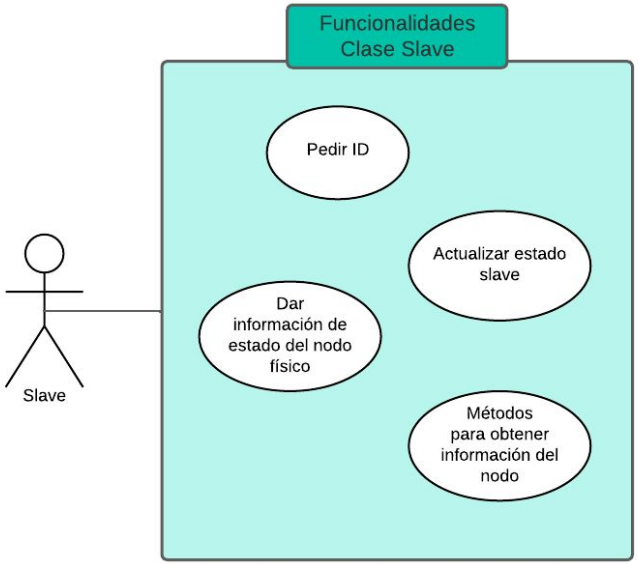


Diagrama de las funcionalidades de la clase slave

Implementación

- Servidor MQTT.
- Librería STRPLibrary.
- **Script python MasterControl.**
- Script python SlaveXControl.
- Script NodeConexCloud.
- Script auxiliar.

Implementación

- MasterControl

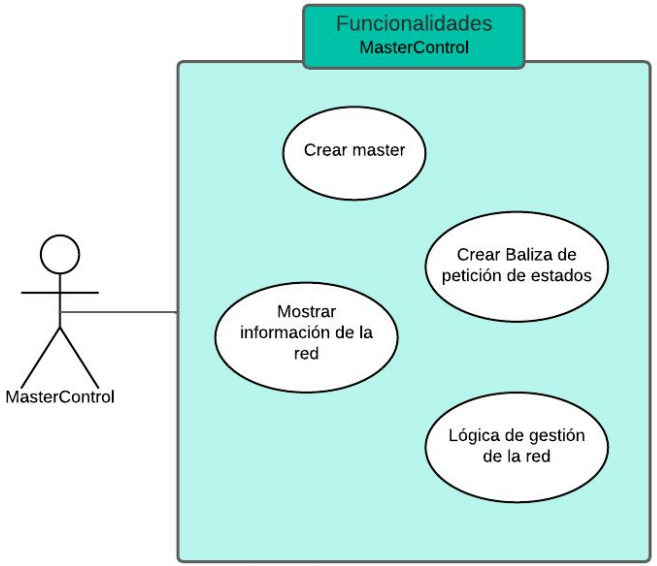


Diagrama de las funcionalidades de MasterControl

Implementación

- Servidor MQTT.
- Librería STRPLibrary.
- Script python MasterControl.
- **Script python SlaveXControl.**
- Script NodeConexCloud.
- Script auxiliar.

Implementación

- SlaveXControl

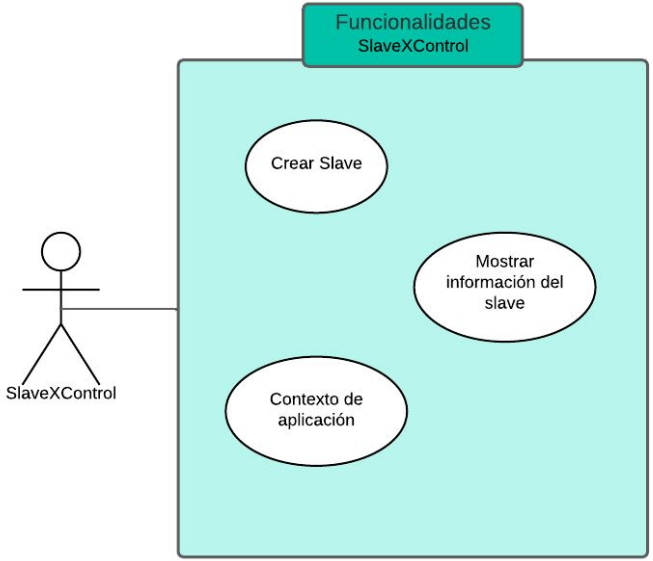


Diagrama de las funcionalidades de SlaveXControl

Implementación

- Servidor MQTT.
- Librería STRPLibrary.
- Script python MasterControl.
- Script python SlaveXControl.
- **Script NodeConexCloud.**
- Script auxiliar.

Implementación

- Servidor MQTT.
- Librería STRPLibrary.
- Script python MasterControl.
- Script python SlaveXControl.
- Script NodeConexCloud.
- **Script auxiliar.**

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- Métodos y materiales.
- Metodología:
- **Resultados y discusión.**
- Conclusión.

Resultados y discusión.

- Ejecución implementación simulada.
- Pruebas.
- Resultados obtenidos.

Resultados y discusión.

- **Ejecución implementación simulada.**
- Pruebas.
- Resultados obtenidos.

Resultados y discusión.

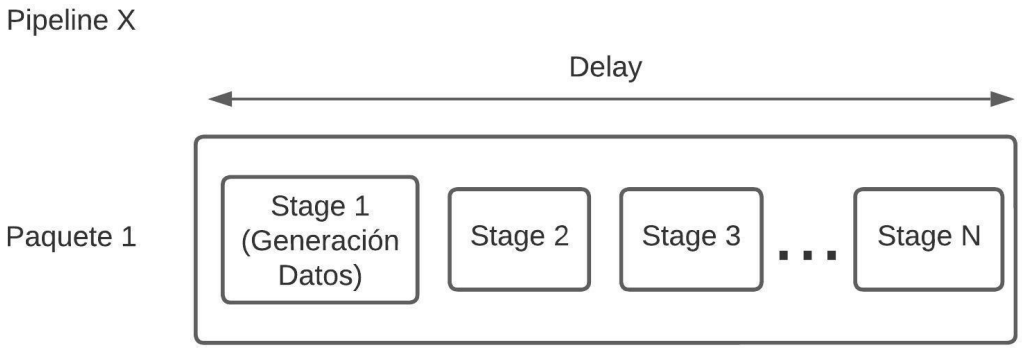
- Ejecución implementación simulada.
- **Pruebas.**
  - **P1** - Uso de otro contexto de aplicación.
  - **P2** - Robustez frente a caídas de nodos.
- Resultados obtenidos.

Resultados y discusión.

- Ejecución implementación simulada.
- Pruebas.
- **Resultados obtenidos.**
  - **Parámetros de medida usados.**
    - **Delay**
    - **Throughput**

Resultados y discusión.

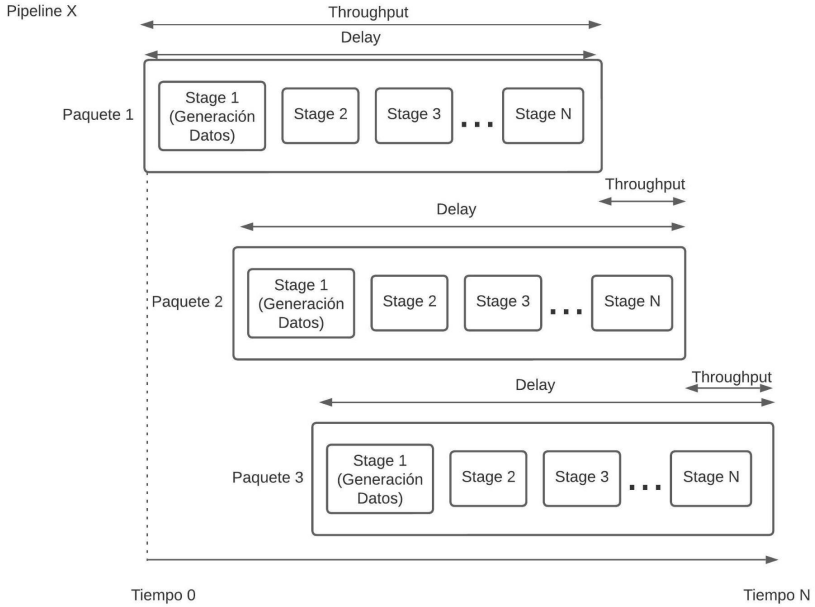
- **Delay**



Delay de un pipeline con sus diferentes stages de procesamiento para el paquete 1

Resultados y discusión.

- **Throughput**



Throughput de un pipeline junto al concepto de delay

Resultados y discusión.

- Ejecución implementación simulada.
- Pruebas.
- **Resultados obtenidos.**
  - **Discusión de los resultados del primer experimento.**
    - 1 solo pipeline
    - 3 stage, generar datos, media y varianza
    - Cada 30 segundos se generan 10 millones de datos

Resultados y discusión.

- **Discusión de los resultados del primer experimento.**

Nº Paquete	Delay	Throughput
0	97.9748	97.9748
1	99.5592	57.7575
2	103.4542	61.978
3	103.8879	57.7149
4	100.9301	60.7902
5	101.8162	57.9822

Pipeline con tres nodos y cada nodo con un stage

Resultados y discusión.

- **Discusión de los resultados del primer experimento.**

Nº Paquete	Delay	Throughput
0	87.7147	87.7147
1	90.3438	57.8115
2	91.3767	59.6288
3	86.787	53.2553
4	88.694	59.9698
5	90.3777	57.8069

Pipeline con dos nodos, el primer nodo con un stage, y el último nodo con dos stages

Resultados y discusión.

- **Discusión de los resultados del primer experimento.**

Nº Paquete	Delay	Throughput
0	72.3517	72.3517
1	72.3735	72.4064
2	72.7274	72.7623
3	72.4742	72.5066
4	74.5678	74.6015
5	73.3515	73.3881

Pipeline con un solo nodo, dicho nodo contiene todos los stages

Resultados y discusión.

- Ejecución implementación simulada.
- Pruebas.
- **Resultados obtenidos.**
  - **Discusión de los resultados del segundo experimento (experimento para caso sintético).**
    - 1 solo pipeline
    - Se crearán dispositivos dependiendo del número de stage
    - En total se probarán de 1 stage a 10 stages
    - Se realizará una media del Delay y throughput de unos 10 paquetes.
    - 10 segundos de procesamiento, siendo el procesamiento de cada stage T/N.

Resultados y discusión.

Nº Stages	Delay	Throughput
1	10.00902	10.01010
2	12.40462	6.05360
3	12.08598	4.53731
4	13.99096	3.95452
5	14.16102	3.84481
6	12.2683	2.95212
7	13.77032	2.93390
8	14.05252	2.85523
9	17.12176	3.38337
10	16.73169	3.10566

Solución Publicador-Suscriptor para Edge/Fog Stream Processing usando Network Pipelining

Índice :

- ¿Por qué se escoge este proyecto?.
- Hipótesis y Objetivos a cumplir.
- Métodos y materiales.
- Metodología:
- Resultados y discusión.
- **Conclusión.**

Conclusión.

Recordemos los objetivos:

- **OB-01-> Desarrollar una metodología para Stream Processing reconfigurable** en entornos locales distribuidos con dispositivos de baja capacidad de cómputo. **LOGRADO!**
- **OB-02-> Determinar la viabilidad** de utilizar tecnologías de comunicación local basadas en el **paradigma publicador-suscriptor** para dar soporte al control de un **Stream Processing Engine**. **LOGRADO!**
- **OB-03-> Evaluar el rendimiento** del desarrollo propuesto en un caso de estudio real. **PARCIALMENTE LOGRADO!**

Conclusión.

Futuras mejoras:

- Crear **contextos de aplicación más variados y más complejos**.
- **Sistema basado en reglas** para la logística del máster.
- **Optimización del árbol de topics** e intercomunicaciones entre dispositivos.
- Completa **división entre el apartado de control y el contexto de aplicación**, usando diferentes brokers de MQTT
- **Mejoras en las pruebas**. Ej: distintos tiempos de ejecución para cada stage.
- Etc

Conclusiones a nivel personal.

Bibliografía:

[1] Karagiannis, V., & Schulte, S. (2020). Comparison of Alternative Architectures in Fog Computing. In 2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC). 2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC). IEEE. https://doi.org/10.1109/icfec50348.2020.00010

[2] Klinaku, F., Zigldrum, M., Frank, M., & Becker, S. (2019). The Elastic Processing of Data Streams in Cloud Environments: A Systematic Mapping Study. In Proceedings of the 9th International Conference on Cloud Computing and Services Science. 9th International Conference on Cloud Computing and Services Science. SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0007708503160323

[3] Garcia, A. M., Griebler, D., Schepke, C., & Fernandes, L. G. (2023). Micro-batch and data frequency for stream processing on multi-cores. In The Journal of Supercomputing (Vol. 79, Issue 8, pp. 9206–9244). Springer Science and Business Media LLC. https://doi.org/10.1007/s11227-022-05024-y

[4] Samza. (n.d.). https://samza.apache.org. https://samza.apache.org/

[5] Apache Flink® — Stateful Computations over Data Streams. (n.d.). Apache Flink. https://flink.apache.org/

[6] Apache SparkTM - Unified Engine for large-scale data analytics. (n.d.). https://spark.apache.org/

Bibliografía:

[7] Zhao, H., Yao, L., Zeng, Z., Li, D., Xie, J., Zhu, W., & Tang, J. (2020). An edge streaming data processing framework for autonomous driving. In Connection Science (Vol. 33, Issue 2, pp. 173–200). Informa UK Limited. https://doi.org/10.1080/09540091.2020.1782840

[8] Gupta, H., & Ramachandran, U. (2018). FogStore. In Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems. DEBS '18: The 12th ACM International Conference on Distributed and Event-based Systems. ACM. https://doi.org/10.1145/3210284.3210297

[9] Yin Liao, Guang-Zhong Sun, & Guoliang Chen. (2009). Distributed Pipeline Programming Framework for State-Based Pattern. In 2009 Eighth International Conference on Grid and Cooperative Computing. 2009 Eighth International Conference on Grid and Cooperative Computing (GCC). IEEE. https://doi.org/10.1109/gcc.2009.11

[10] IBM Developer. (s. f.). https://developer.ibm.com/podcasts/ibm_developer_podcast/034-mqtt-inventor-uk-ireland-cto-andy-stanford-clark/

[11] Vitorino, J. P., Simão, J., Datia, N., & Pato, M. (2023). IRONEDGE: Stream Processing Architecture for Edge Applications. In Algorithms (Vol. 16, Issue 2, p. 123). MDPI AG. https://doi.org/10.3390/a16020123

[12] Yao, L., Zhao, H., Tang, J., Liu, S., & Gaudiot, J.-L. (2021). Streaming Data Priority Scheduling Framework for Autonomous Driving by Edge. In 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC). 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE. https://doi.org/10.1109/compsac51774.2021.00017

¡Muchas gracias por su atención!