Estudio e implementación de una técnica de clustering dinámico para trabajar con flujos de datos

Tesina de Licenciatura en Informática.
Alumno: Molina Roberto Pedro.
Director: Dr. Waldo Hasperué

RoadMap

1. Introducción
2. Flujos de datos
3. Streaming clustering: State of the Art
4. D3CAS
5. Conclusiones

Objetivos

- **Estudiar y analizar** las técnicas y problemáticas existentes de **clustering** (agrupamiento) aplicadas sobre los **flujos de datos**, buscando técnicas de agrupamiento dinámico.
- **Investigar y estudiar** el modelo de flujos datos y los frameworks o plataformas de procesamiento de flujos de datos actuales con el fin de analizar la viabilidad para generar técnicas de clustering sobre estos entornos.
- **Desarrollar e implementar** un algoritmo de clustering dinámico aplicado al tratamiento de flujos de datos, junto con las evaluaciones y comparaciones correspondientes de los resultados obtenidos.

¿ Clustering . . .

. . . Flujos de Datos ?

El Contexto.

Situación donde un sistema ‘X’ recibe información que **no** puede **almacenar**, debido a que llega constantemente grandes cantidades de datos que puede **sobrepasar** la capacidad de **almacenamiento** del sistema en un breve lapso de tiempo.

Por lo tanto, bajo esta situación, se deben procesar dichos datos ni bien llegan al sistema e ir descartando los datos ya procesados para seguir analizando los restantes.

Características.

1. Toda la información se encuentra **online** y **cambia** de forma continua en el tiempo.
2. El sistema **no tiene control** sobre el orden en el que los elementos de datos llegan para ser procesados.
3. Tienen un tamaño potencialmente **ilimitado**, lo que lleva a que no se puedan almacenar en su totalidad.
4. Deben ser procesado de manera '**on-the-fly**'.
5. Un elemento procesado, se **descarta** o **archiva**.

Definición Flujos de datos.

Los **flujo de datos** o ‘**datastream**’ son una secuencia de elementos en tiempo real, continua, ordenada por timestamp, variables en el tiempo y posiblemente impredecibles e ilimitados, debido a que es imposible controlar en qué momento llegan los elementos, y tampoco es posible almacenar localmente una secuencia en su totalidad.

Ejemplos de flujos.

- **Redes Sociales:** Twitter, Facebook, Instagram.
- **Flujos de transacciones:** Bancarias o criptomonedas (Bitcoin).
- **Monitoreo de redes:** Detección de intrusiones en la red, logs de servidores.
- **Monitoreo en tiempo real de sensores** + **Internet de las Cosas** (IoT).
- **Análisis climáticos.**
- Análisis de información generada por **dispositivos wearable.**

Desafíos en Flujos de datos.

- **Single-pass:** procesamiento en una única lectura, debido a que no se puede ni almacenar el flujo en su totalidad ni recuperar datos ya recibidos.
- **Tiempo real:** el procesamiento de un lote de datos debe ser rápido, de lo contrario, no se podrá seguir la velocidad del flujo.
- **Memoria limitada:** los datos originales se descartan, entonces se debe generar información que mejor aproxime el estado del flujo.
- **Concept-drift:** Se debe detectar los posibles cambios de distribución de los datos que pueden aparecer en el tiempo.

Ventanas de tiempo.

Flujos potencialmente **infinitos**, por lo que el flujo se divide y procesa por partes. A la **porción** de datos actual sobre la cual vamos a trabajar se la define como **ventana temporal** de elementos:

$$W[i,j] = (x_i, x_{i+1}, x_{i+2}, \ldots, x_j)$$

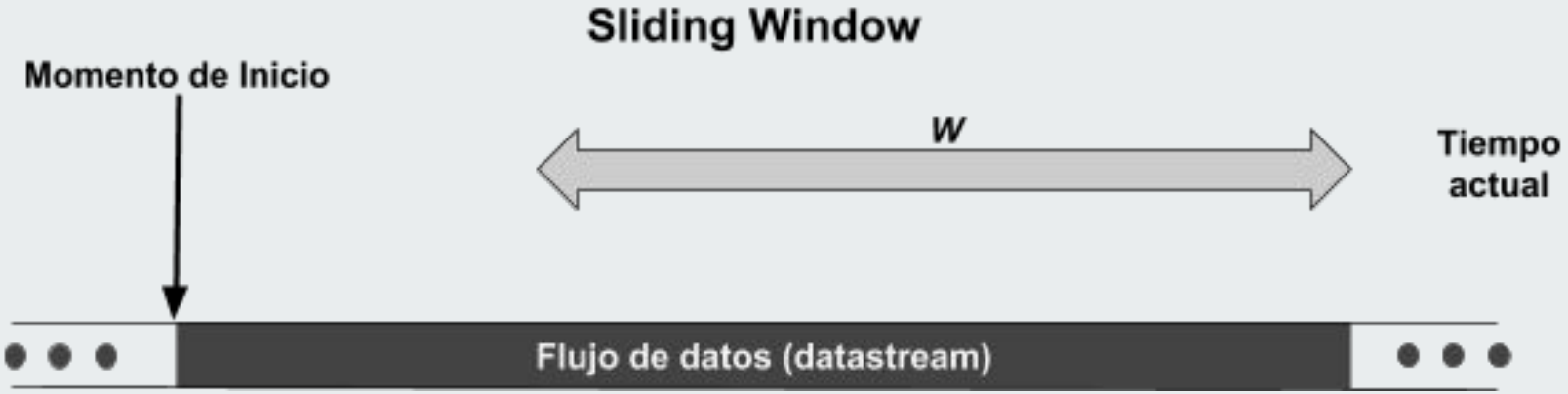
donde *W[i, j]* representa la ventana que contiene los elementos del flujo que se encuentran entre los puntos en el tiempo *i j* . Se cumple i<j.

Tipos de ventanas.

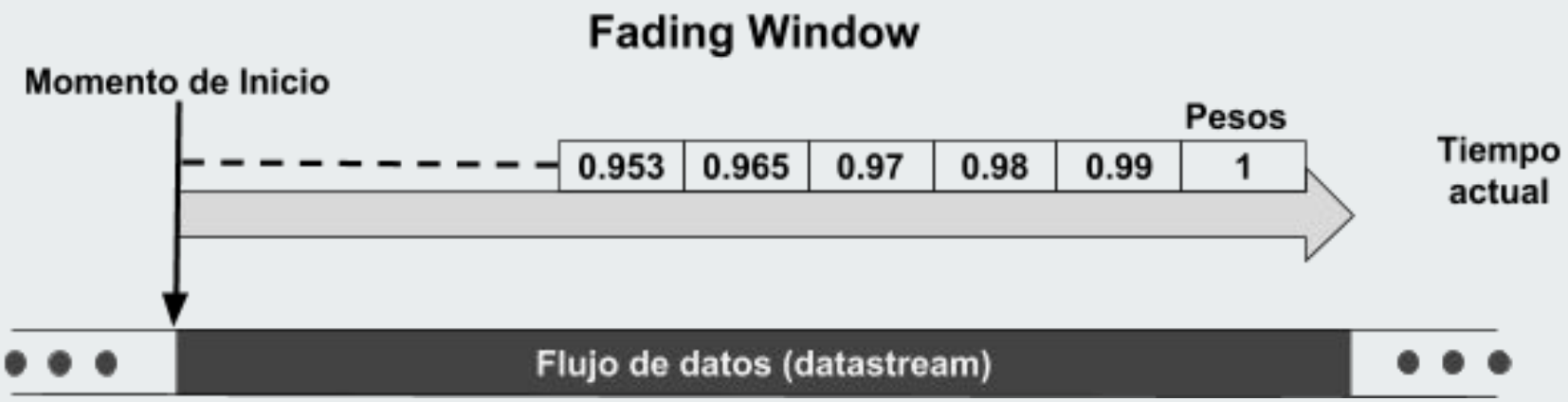
Existen 4 tipos populares de ventajas temporales:

- Landmark Window
- Sliding Window
- Fading Window (Damped Window)
- Title Time Window

Tipos de ventanas.



Tipos de ventanas.



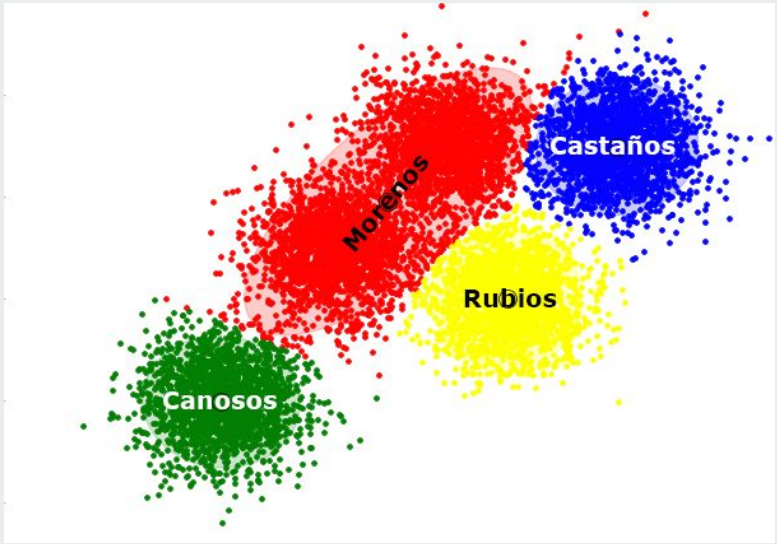
Flujos de Datos ✓ ✓

. . . pero cómo los procesamos ??

Respuesta: Machine learning, Data Mining, aprendizaje supervisado, no supervisado, Clustering !!

Clustering.

- Clustering, o **técnica de agrupación**, es una técnica de **Minería de datos.**
- Es una técnica de aprendizaje **no supervisado**.
- Popular para realizar análisis estadísticos.
- La técnica consiste en dividir y agrupar automáticamente un conjunto de datos en grupos de datos similares.



Características de clustering.

- **Permitir clasificar objetos** en grupos de objetos similares.
- **Reducir la cantidad de datos** por medio de un **modelo** que se encargue de representar las características generales del grupo.
- **Buscar patrones** en el conjunto de datos.

#Algortimos de clustering.

- Floyd (K-Means)
- K-Means ++
- Mean-Shift Clustering
- DBSCAN
- Clustering aglomerativos
- Spectral Clustering

Flujos de Datos ✓ ✓

Clustering ✓ ✓

Estado del Arte Clustering en Flujos de datos

Streaming Clustering

BIRCH

CluStream

ClusTree

DenStream

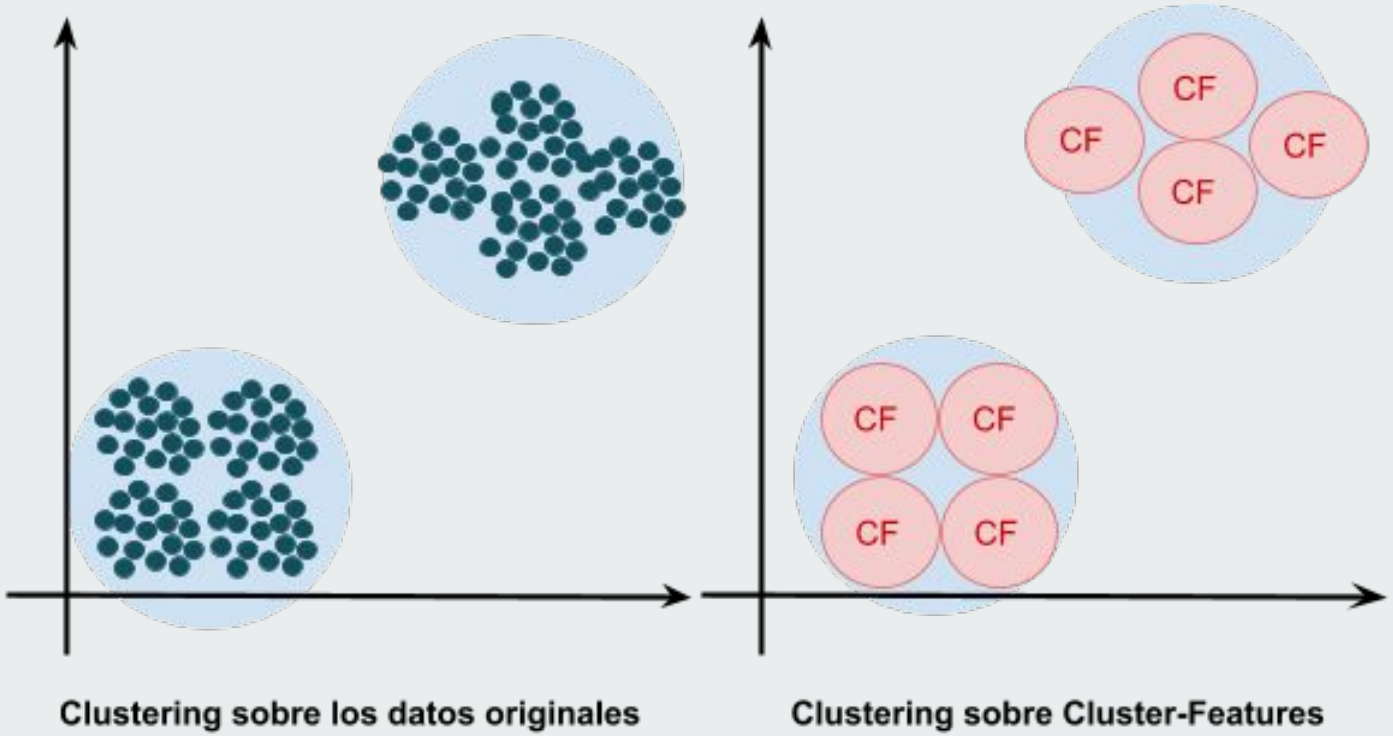
BIRCH características.

- Procesamiento en un sólo “**scan**” de los datos ⇒ única lectura de datos.
- Propone generar **Resúmenes** de los datos, llamados **Cluster Feature**.
- **CF** Representa un subconjunto de N datos presentes en un flujo.
- Luego, aplica técnicas de Clustering directamente sobre los **Cluster Feature** y no sobre los datos originales.
- Fue diseñado para trabajar con dataset con un volumen masivo de información (**BIG DATA**).
- Los CF se almacenan en un Árbol de tipo B.

BIRCH Cluster Feature.

- Se modela como una 3-tupla ⇒ **CF (N, LS, SS)**
  - **N**: es el número total de datos presente en el subconjunto.
  - **LS**: Suma lineal de los **N** elementos en el subconjunto.
  - **SS**: Suma cuadrática de los **N** elementos en el subconjunto.
- Con **N, LS, SS**, se puede calcular todas las propiedades y medidas necesarias para aplicar clustering como por ejemplo **centro** y **radio**.

BIRCH Cluster Feature.



BIRCH Resumen.

Características importantes

- Primer algoritmo en proponer almacenar información resumida.
- Puede manejar una gran cantidad de datos y detectar outliers.
- Luego de generar el árbol CF, se puede aplicar cualquier algoritmo de clustering sobre los elementos que contiene. (ADAPTACIÓN)

Limitaciones

- Para mejorar la calidad del agrupamiento se deben realizar varias iteraciones.
- Problemas heredados de la estructura de Árbol (2 CF similares se encuentran en distintos nodos, 2 CF que no deberían estar juntos forman parte del mismo nodo)
- No detecta Concept Drift.

CluStream características.

- **Micro-Clusters** ⇒ extiende el concepto de Cluster Feature agregando información relacionada al timestamp de los datos.
- Enfoque 2 fases ⇒ **Online-Offline**
- Landmark Window.
- **Snapshots** ⇒ respaldo Micro-Clusters que se generaron en distintos instantes de tiempo.
- KMeans
- La **inicialización** de los micro-clusters se lleva acabo utilizando espacio de disco y ejecutando kmeans por única vez.

CluStream Online-Offline.

Clustream es popular por definir el enfoque online-offline sobre streams. Todo el proceso de clustering se divide en 2 componentes:

- **Online**: proceso encargado de generar y mantener Micro-Clusters.
- **Offline**: proceso que consumen los Micro-Clusters para proporcionar los resultados de clustering global a los usuarios.

Ambas etapas puede trabajar **simultáneamente**. Ampliamente utilizado en muchos algoritmos debido a su eficiencia en el manejo de los flujos.

CluStream Resumen.

Características importantes

- Primer algoritmo en proponer metodología Online-Offline.
- Nuevo enfoque de micro-clusters temporales.
- Permite evaluar el flujo en diferentes intervalos de tiempo (COSTOSO)
- Fácil de entender e implementar.

Limitaciones

- **Redundancia** de micro-clusters.
- Uso de **DISCO**.
- Problemas heredados de algoritmo de clustering por partición.
- No detecta clusters dinámicamente.

ClusTree.

- Metodología **Online-Offline**.
- Utiliza **Micro-cluster** al igual que Clustream, pero son ponderados/discriminado por tiempo de llegada.
- Utiliza la idea de Árbol de CF pero utiliza un Árbol R*.
- Se adapta a la velocidad del flujo utilizando buffer.
- Cada nodo en el árbol contiene un **CF** + **Buffer**.
- Concepto **anytime insertions**.

ClusTree Anytime Insertions.

- A velocidades de flujo **altas**, se produce que queden elementos sin procesar y por ende se pierden, debido al tiempo invertidos en las acciones de búsqueda e inserción.
- ClusTree **interrumpe** las inserciones cuando hay altas velocidades, guardando los datos en el buffer del nodo alcanzado en la búsqueda.
- El próximo proceso de inserción que llegue al nodo intermedio, **recuperará** los elementos, continuará el proceso de inserción de estos junto con el que se encuentra llevando a cabo.

ClusTree Resumen.

Características importantes

- Mejora la calidad de los clusters finales en flujos de alta velocidad.
- Mejor utilización del tiempo gracias al uso de inserciones Anytime.
- Mantiene micro-clusters que representan datos recientes, por ende los clusters globales siempre están actualizados.

Limitaciones

- Problemas con outliers.
- Complejidad alta en implementación, poco escalable ya que genera mucho overhead para mantener actualizado tanto los micro-clusters como los buffers.
- Mayor consumo de memoria por el uso de buffers.

DenStream.

- Ventana de tiempo fading windows.
- Utiliza CF o **Micro-cluster** al igual que Clustream, pero son ponderados/discriminado por tiempo de llegada.
- Metodología **Online-Offline**.
- Clustering basado en **densidad**.
- **Detección dinámica** de clusters.
- Detección de clusters con **formas arbitrarias**.
- **Discriminación** de micro-clusters y micro-cluster outliers.
- No utiliza estructura de Árbol.

DenStream Resumen.

Características importantes

- Define una estrategia para filtrar outlier en la etapa online.
- Detección de concept-drift a partir de micro-clusters “pesados”.
- Gracias a DBSCAN detecta un número dinámico de cluster y de forma arbitraria.

Limitaciones

- Actualizaciones de micro-clusters más lentas.
- Uso de buffer en disco para outlier.

Conclusiones del estado del arte.

Algoritmo ideal de clustering:

1. Procesamiento en una sola iteración.
2. Bajo uso de memoria y CPU.
3. Detección dinámica: No conocer a priori ni cantidad ni formas de los grupos.
4. Filtrar ruido - outliers.
5. Descubrir agrupaciones sobre distintas ventanas de datos.
6. Representación compacta de los datos.
7. Resultados en near-real-time.
8. Detectar agrupaciones sin utilizar parámetros asignados por el usuario.
9. Habilidad de mantener agrupaciones para distintos puntos en el tiempo.
10. **Procesar sobre una arquitectura distribuida (escalabilidad).**

Problemas con estos algoritmos:

No Escalan!

Apache Spark.

- Sistema de **cómputo distribuido** de alto rendimiento y de propósito general diseñado para procesamiento sobre un cluster/conjunto de PCs.
- Motor más **popular** tanto en entornos académicos como en la industria.
- Extiende el modelo popular de **MapReduce**, adaptando su diseño y API con las características del framework distribuido **DryadLINQ**.
- **Velocidad** de Ejecución.
- Evaluación **Lazy**.
- Proyecto más activo de Apache con + 1.000 colaboradores.

Apache Spark Arquitectura.

- Arquitectura maestro/esclavo (master/slave).
- Un Coordinador central denominado **master** o **driver**.
- N nodos trabajadores distribuidos denominados **executors** o **ejecutores**.
- Tanto el master como cada executor ejecutan procesos Java separados.
- Master + Executors = Aplicación Spark.
- El master contiene el programa escrito por el **Usuario**.

Apache Spark API.

- Spark brinda una **API** sencilla de alto nivel.
- Los **usuarios** escriben **programas secuenciales** utilizando operadores imperativos o declarativos sobre los datos.
- Spark automática y transparentemente se encarga de **transformar**, **optimizar** y **compilar** dichos programas en **tareas distribuidas** con una eficiente **paralelización de datos** sobre un clusters de PCs.
- Las tareas resultantes son diseñadas para trabajar bajo el modelo MapReduce.

Apache Spark API.

- **Transformaciones**: operaciones ejecutadas en los workers y generan nuevos datos a partir de modificar, filtrar o procesar los datos de entrada.
- **Acciones**: Devuelven los datos procesados al nodo master.

Cada programa Spark debe contener una acción, ya que las acciones devuelven información al controlador master, las acciones son lo que fuerzan la evaluación de un programa Spark, por lo tanto, siempre son las encargadas de forzar la inicialización del programa.

D3CAS

Distributed Dynamic Density based Clustering Algorithm for dataStream

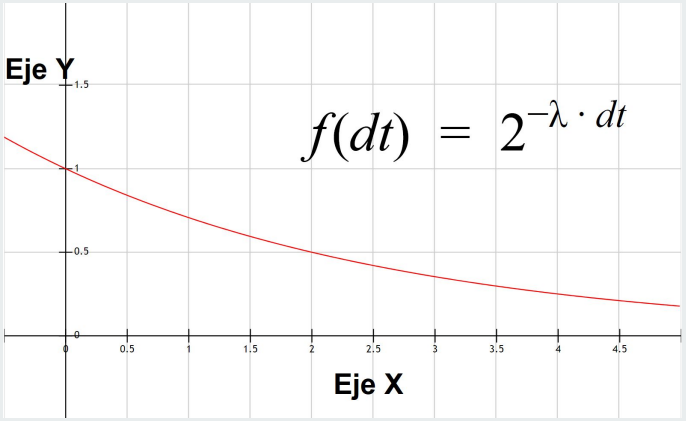
Algoritmo de agrupamiento dinámico distribuida basado en densidad para flujos de datos

D3CAS Características.

- **Detección dinámica** del número de clusters presentes en el flujo.
- Detección de clusters con formas arbitrarias.
- Algoritmo **distribuido** de clustering corriendo sobre **Spark Streaming**.
- Detección de **Concept Drift**.
- Detección de **Ruido**.
- Procesamiento en 2 Fases: **Online-Offline**.
- Fading Window + Micro-Clusters pesados.

D3CAS Ventana de tiempo.

- Modelo de ventana **Fading Window** o **Damped Window**.
- Los datos almacenan un atributo que determine su importancia en el tiempo, lo cual facilita determinar qué datos representan la información más reciente.
- Función de desvanecimiento similar a **DenStream** y **ClusTree**



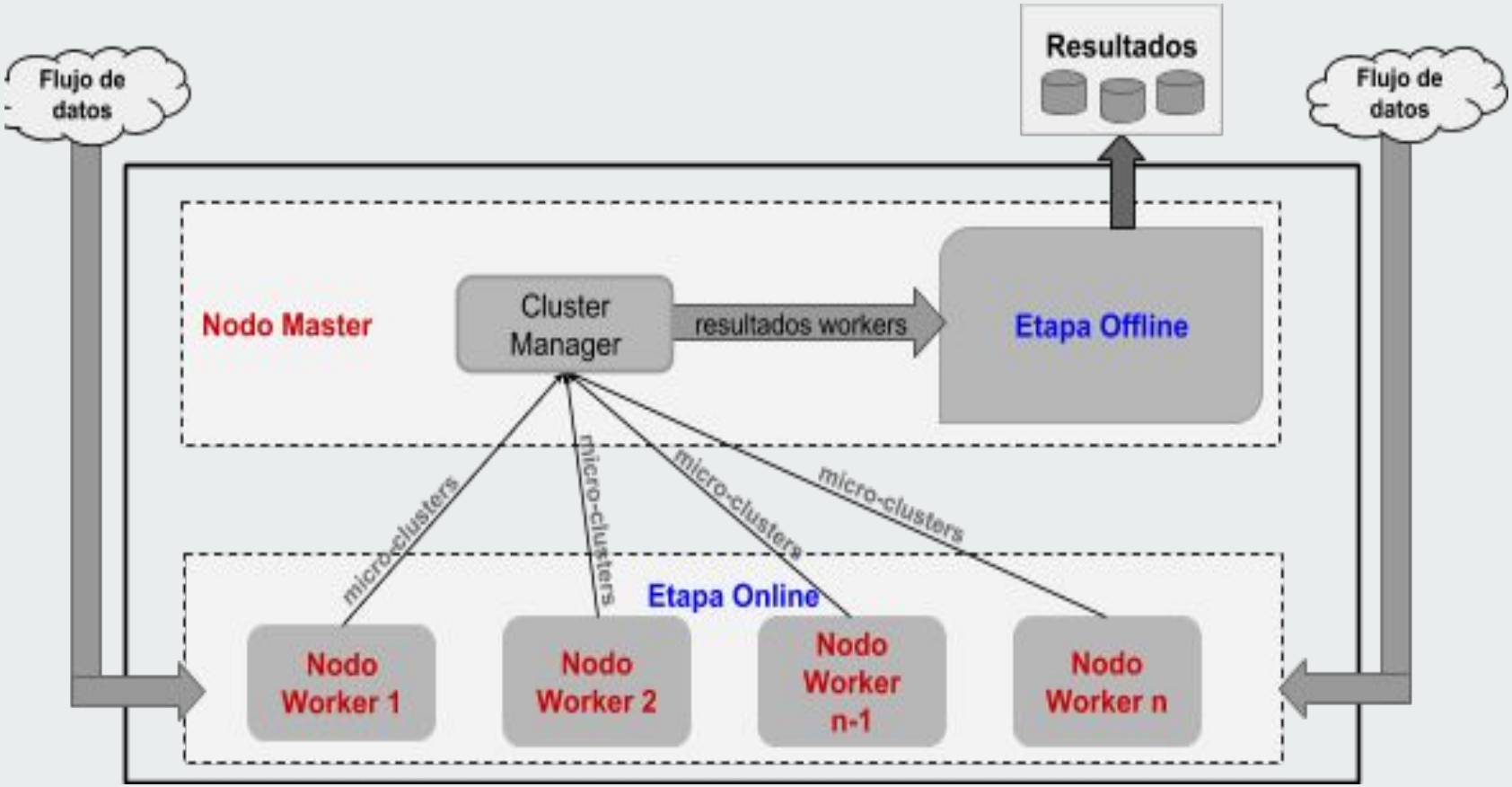
Donde **dt** representa la diferencia entre el timestamp de llegada de un dato y el timestamp del instante de tiempo actual (o un timestamp mayor al de llegada). La constante **λ** determina la importancia que se le da a los datos en comparación con los datos más recientes. A mayor valor de **λ** menor importancia a los datos viejos.

D3CAS Online-Offline.

El desafío recae en diseñar la lógica de la metodología Online-Offline sobre el modelo de Spark (master/workers).

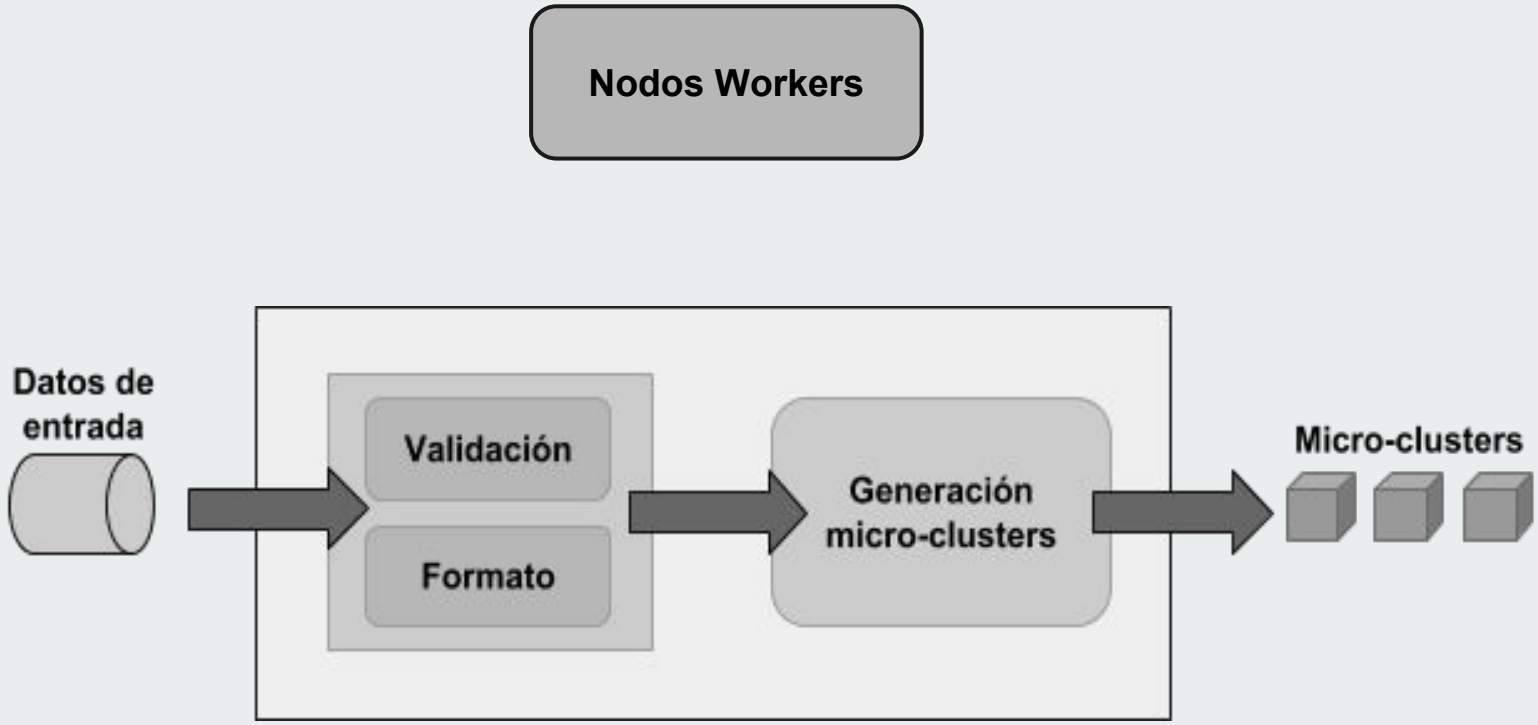
- La etapa **Online** se ejecuta paralelamente en los nodos workers, la tarea de generar los micro-clusters se ajusta al tipo de procesamiento que realizan las operaciones de **transformación** de Spark.
- La etapa **Offline**, se ejecuta en el nodo Master, para realizar clustering se necesita conocer todos los micro-clusters generados. Esto se ajusta al modelo de una operación de **Acción** de Spark, primero se juntan todos los micro-clusters y luego generar un resultado a partir de estos.

D3CAS Online-Offline.



Diseño Online-Offline sobre Spark

D3CAS Online.



D3CAS Micro-cluster.

Un **Micro-Cluster** en un instante de tiempo *t* para un subconjunto **P** de **n** puntos $p_1, ..., p_n$ de **d** dimensiones con sus respectivos timestamp $T_1, ..., T_n$ se define como una 7-tupla ⇒

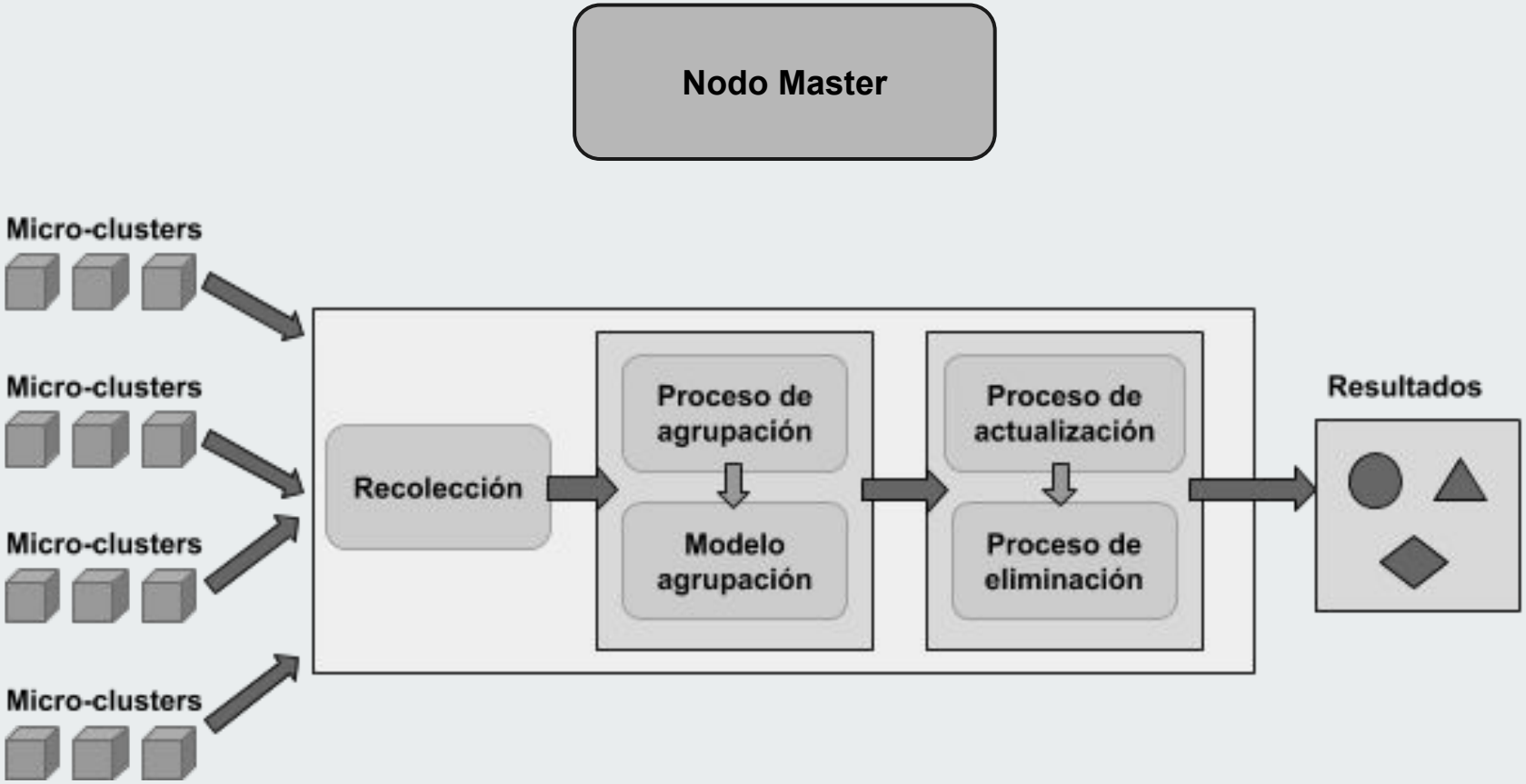
MC (**LS**, **SS**, **n**, **w**, **d**, t_c, t_m) donde

- **LS**, **SS** y **n** iguales que en BIRCH.
- **w** representa el peso o importancia en el tiempo del micro-cluster.
- **d** representa la dimensionalidad de los puntos de **P**.
- t_c representa el timestamp de creación del micro-cluster.
- t_m representa el timestamp de la última actualización del micro-clusters.

D3CAS Generación de micro-clusters.

- Proceso basado en la propiedad de distancia entre puntos.
- No se aplica clustering en esta etapa.
- Consiste en reducir la cantidad de datos, agrupándolos en áreas.
- Cada área representa grupo de datos del flujo, donde la distancia euclidiana entre los datos no supera el valor **e.**
- Luego utilizando los grupos formados, se generan los micro-clusters que se encargan de representar el área.
- Los micro-clusters luego serán enviados al nodo master.

D3CAS Offline.



D3CAS Clustering.

- Agrupación basada en **densidad** ⇒ DBSCAN original.
- Permite detectar **agrupaciones dinámicamente** sin conocer de antemano la cantidad exacta de agrupaciones que pueden existir en el conjunto de datos.
- Permite detectar clusters con formas arbitrarias.
- Permite detectar elementos que representan valores atípicos.
- **Puntos virtuales**: Se utiliza el centro de los micro-clusters para representar los datos de entrada de DBSCAN.
- Utilizando la idea de punto virtual se puede utilizar cualquier técnica de clustering.

D3CAS Actualización temporal.

- Diferente a Clustream y Denstream.
- La actualización del peso de los micro-clusters se realiza en la fase offline. Ya que los micro-clusters siempre se encuentran en el nodo Master y los Workers solo tienen los micro-clusters generados a partir de los datos recién llegados al flujo.
- La actualización de los micro-clusters se realiza periódicamente, y específicamente se actualizan los atributos de peso **w** y los atributos **LS** y **SS**.
- Para la actualización se utiliza la función de enjevecimiento definida por el modelo de ventana de tiempo.

D3CAS proceso de eliminación.

- Luego del proceso de actualización.
- Se eliminan los micro-clusters que se consideran viejos.
- Un micro-cluster se considera viejo y debe ser descartado si su peso w es menor al umbral μ .
- El umbral μ es un parámetro global de esta técnica, por lo que el usuario es el responsable de asignar un valor adecuado dependiendo del contexto.
- Si μ es un valor muy cercano a 1, entonces sólo se conservarán los datos más recientes en el flujo, y cuando sea más cercano a 0, más grande será la ventana de tiempo representada por los micro-clusters.

	BIRCH	Clustream	Clustree	DenStream	D3CAS
Enfoque	online-offline	online-offline	online-offline	online-offline	online-offline
Estructura	feature vectors	Tree-B micro-cluster	Tree-R micro-cluster	micro-cluster	micro-cluster
Ventana	Landmark	Landmark	Damped	Damped	Damped
Algoritmo clustering	K-means	K-means	K-means	DBSCAN	DBSCAN
Detección formas esféricas	✔	✔	✔	✔	✔
Detección formas arbitrarias			✔	✔	✔
Detección ruido				✔	✔
Concept Drift			✔	✔	✔
Distribuido					✔

Evaluación y comparación

Evaluación de clusters.

- Criterios internos: Silhouette
- Criterios externos: Pureza
- Criterios relativos: Comparación con otras técnicas.

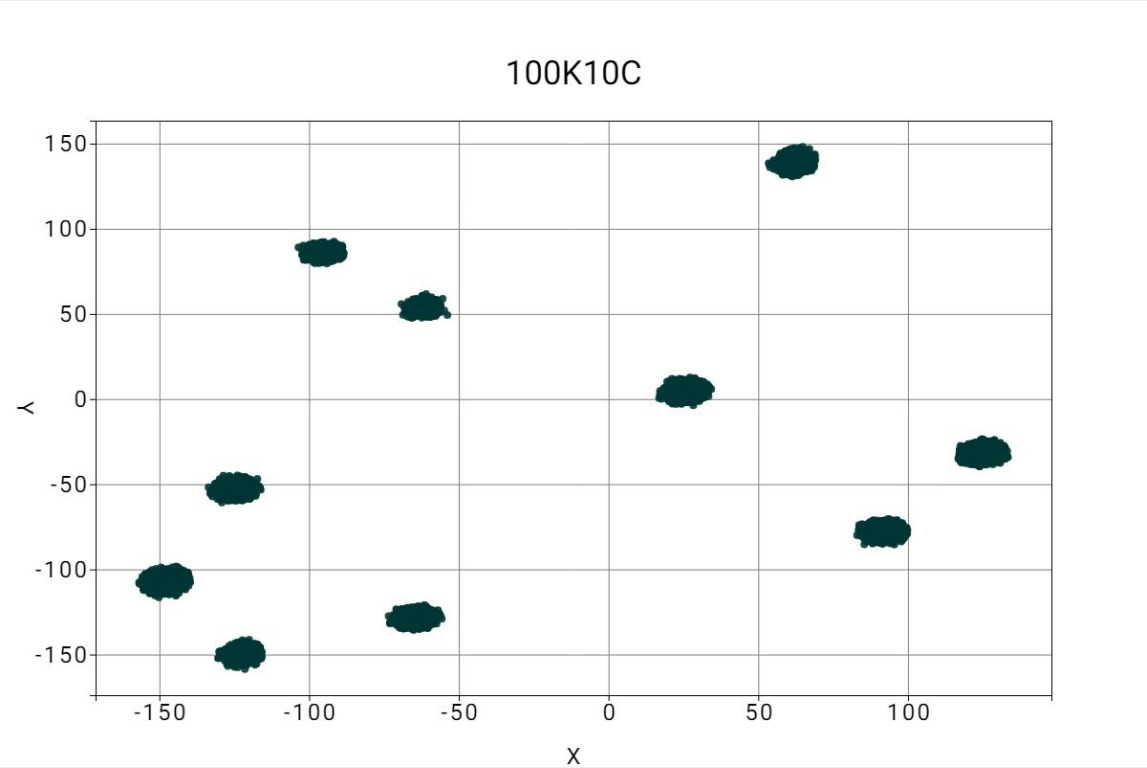
Evaluación Entorno de pruebas.

- Los experimentos fueron ejecutados en un entorno local, es decir, en una única computadora, no en un clusters.
- El objetivo de los experimentos se centra en evaluar la calidad de los resultados obtenidos y no en hacer un análisis de rendimiento y eficiencia.
- Para representar los Flujos de datos, se implementó un simulador que consumen datasets sintéticos de clustering.
- El simulador genera un flujo de datos particionando el dataset y enviando cada partición en distintos instantes de tiempo.

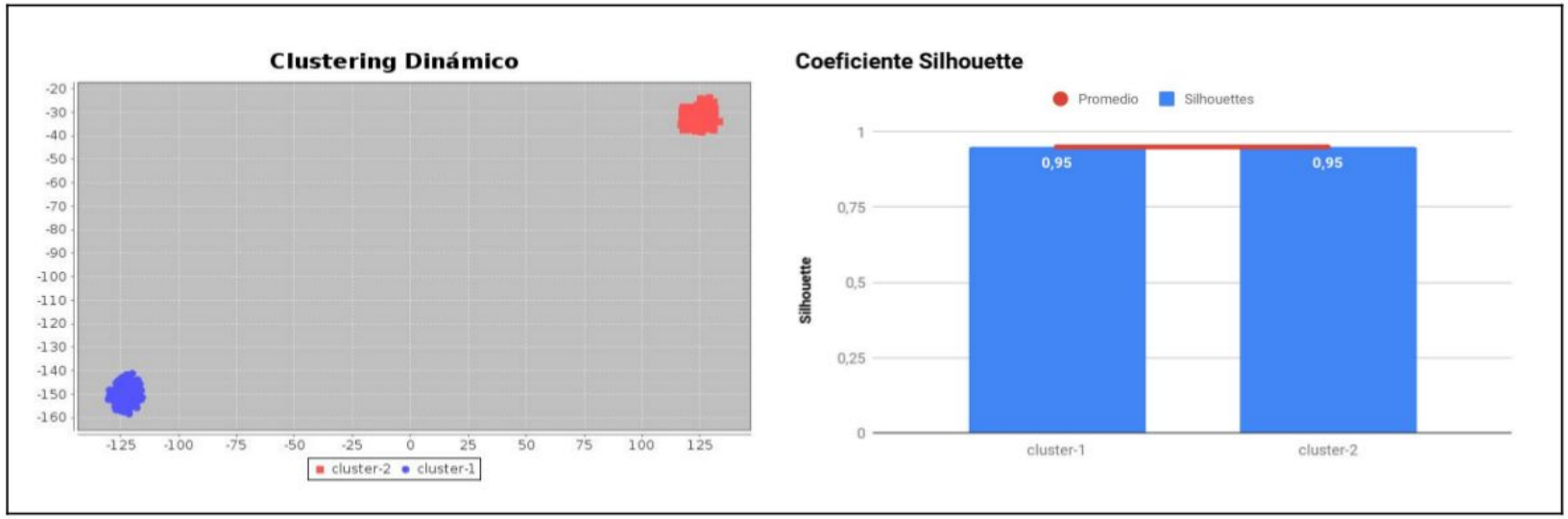
Evaluación de clusters.

Criterio interno: Silhouette + **Detección dinámica** de clusters.

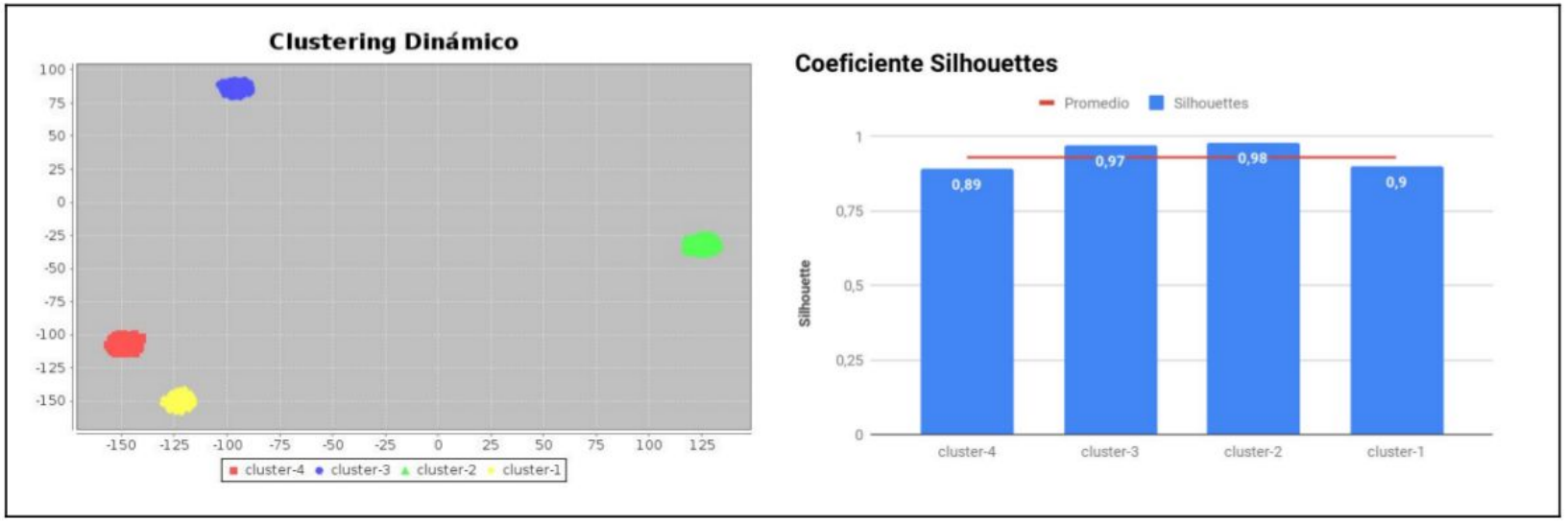
Flujo de 100.000 elementos. Velocidad del Flujo: 10.000



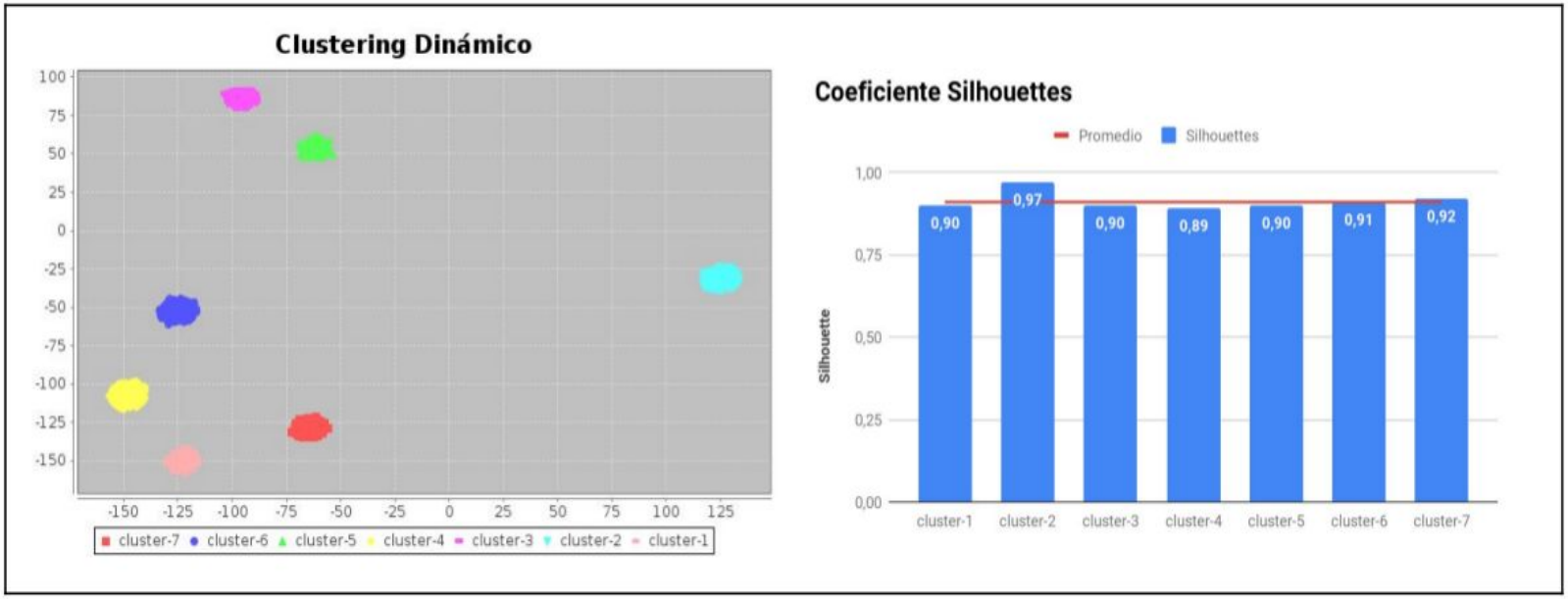
Silhouette
0,95



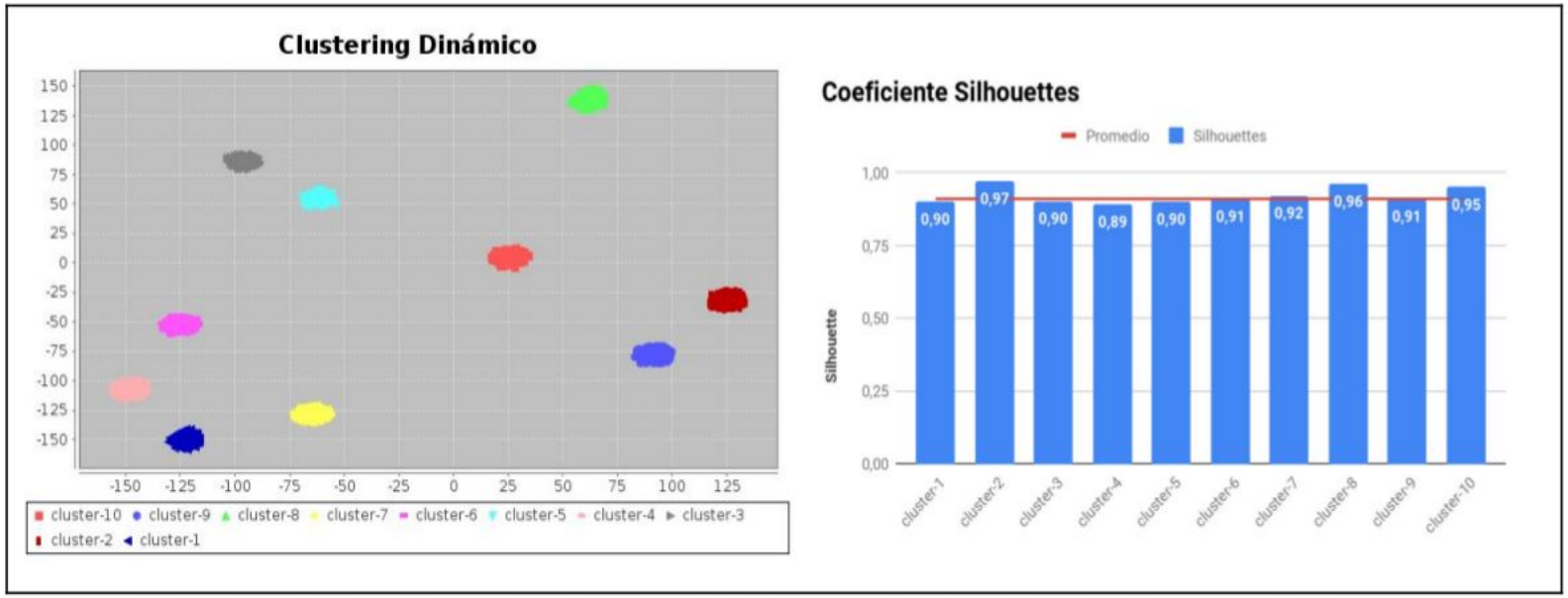
Silhouette
0,93



Silhouette 0,91



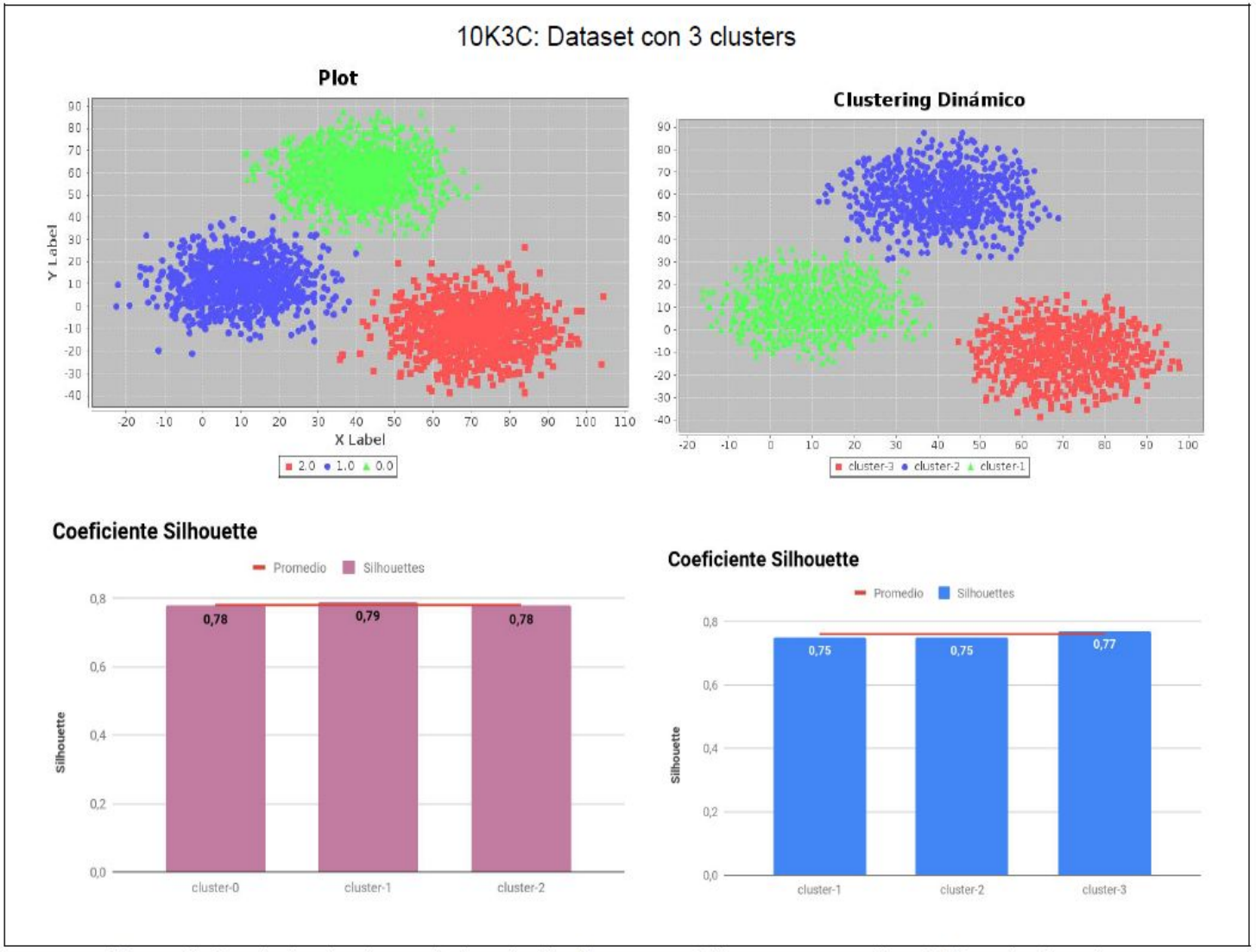
Silhouette 0,91



Comparación.

- Silhouette + comparación contra Clustream.
- Implementación de Clustream que se ejecuta sobre Spark.
- Clustream desarrollada por el proyecto **StreamDM** de la compañía Huawei.

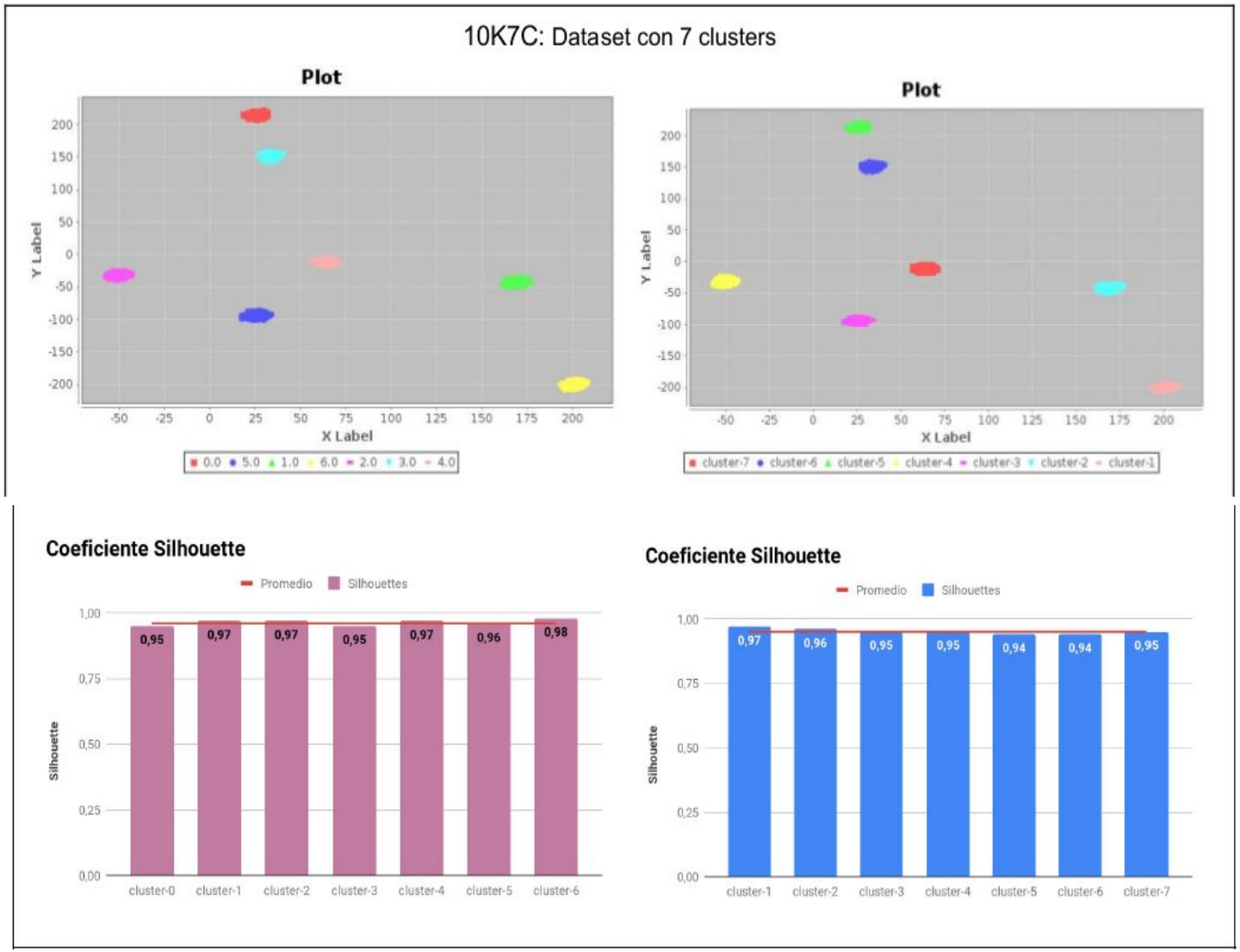
Izquierda: Clustream - Derecha: D3CAS



Silhouette 0,78

Silhouette 0,76

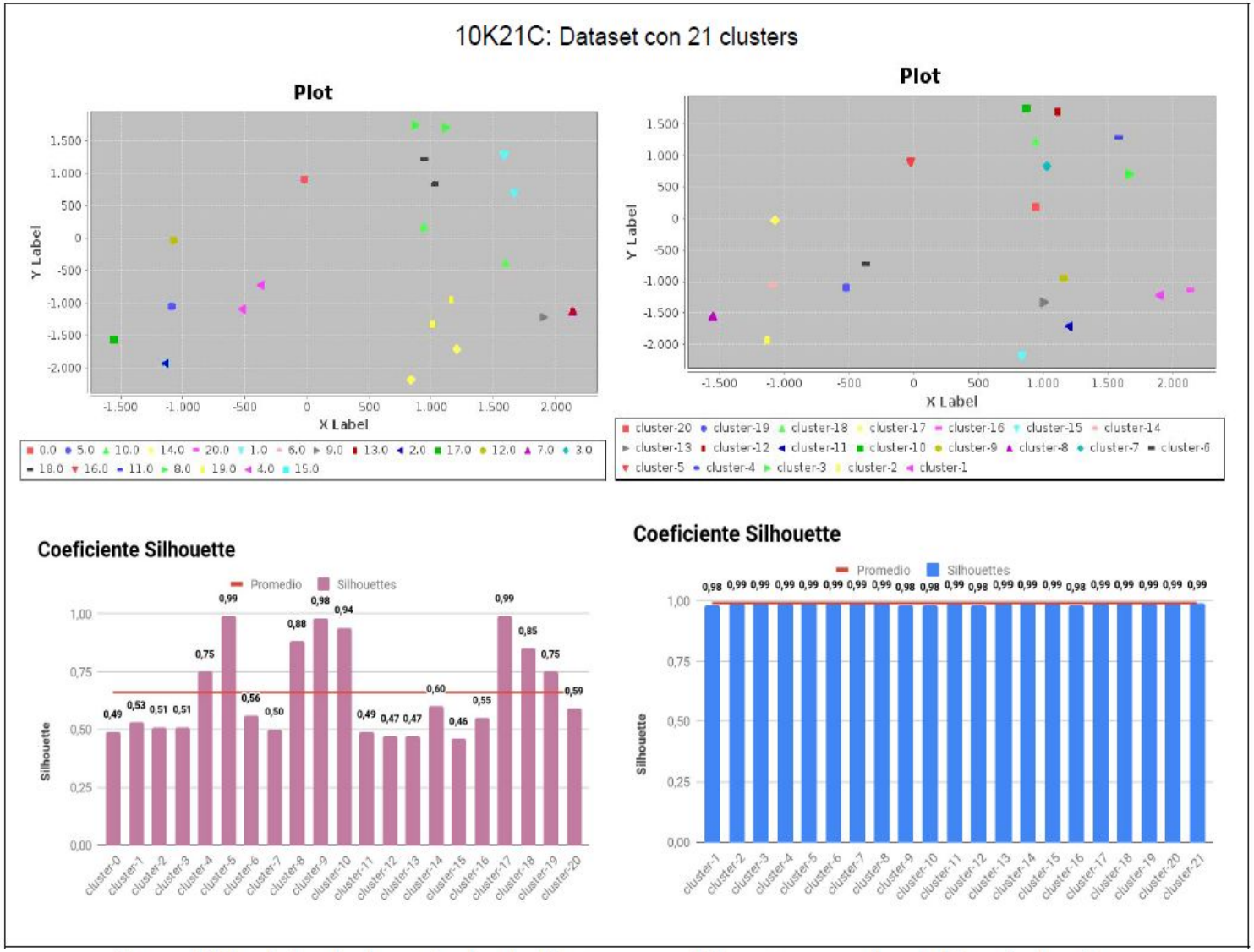
Izquierda: Clustream - Derecha: D3CAS



Silhouette 0,96

Silhouette 0,95

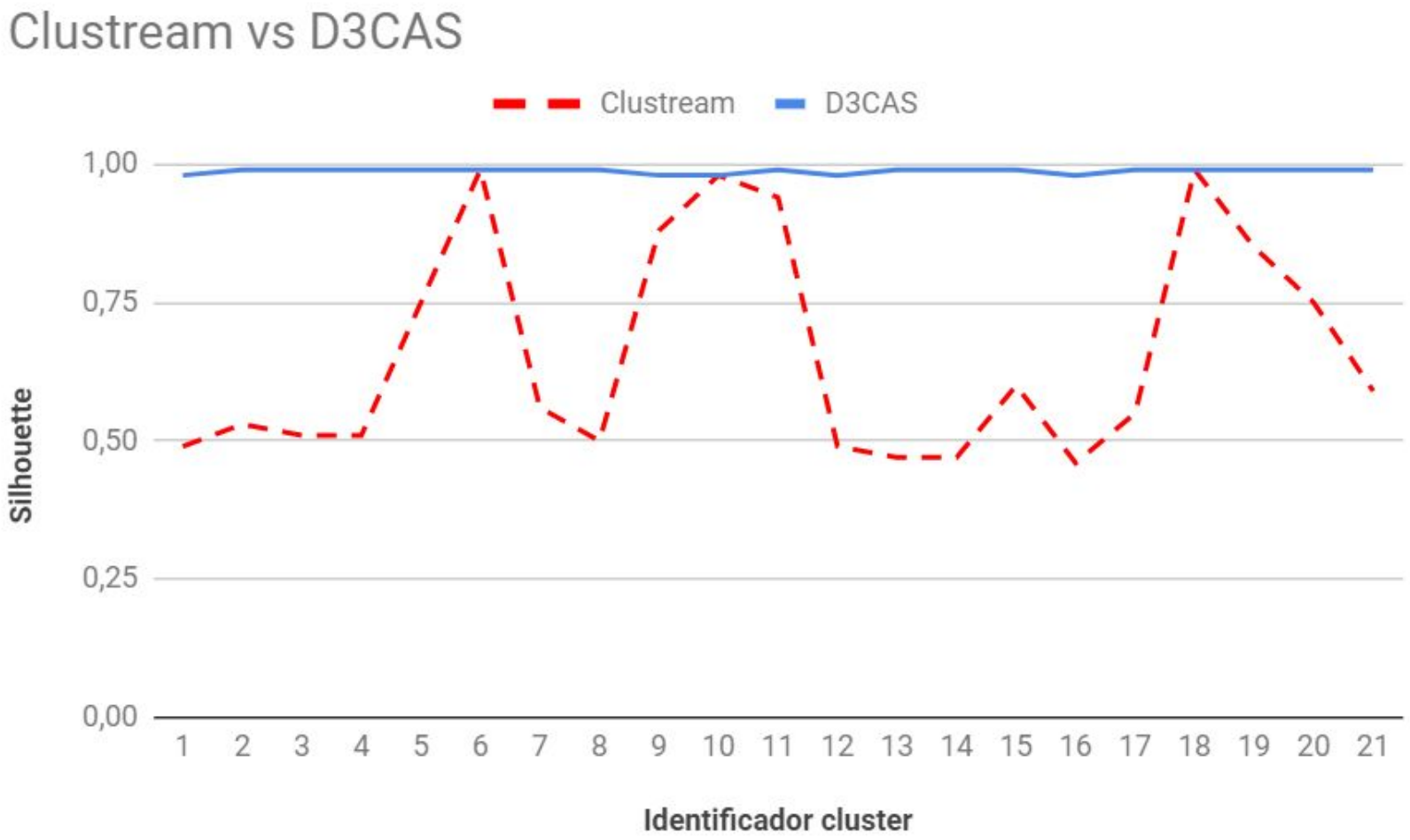
Izquierda: Clustream - Derecha: D3CAS



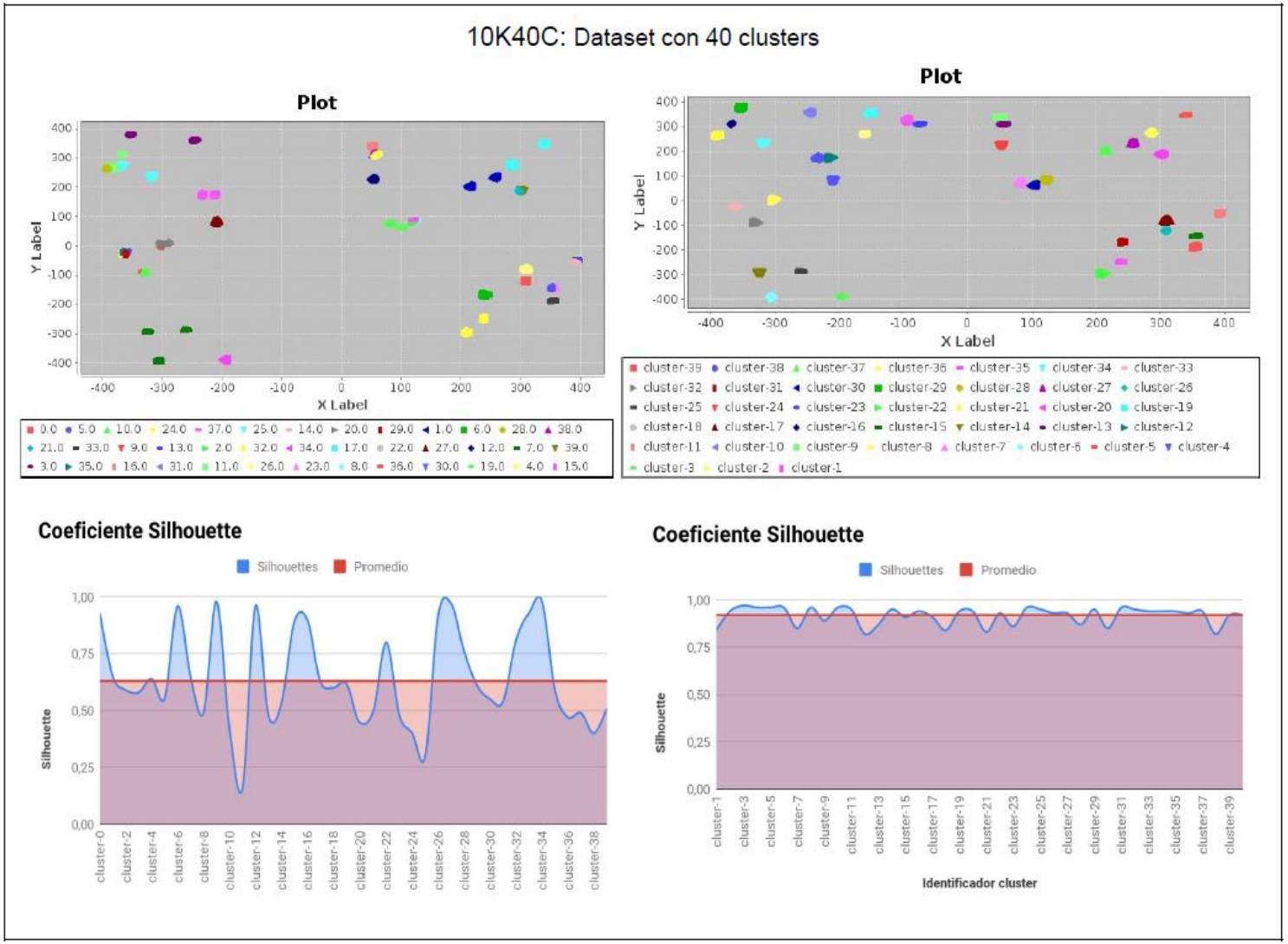
Silhouette 0,66

Silhouette 0,98

Dataset 21 Clusters



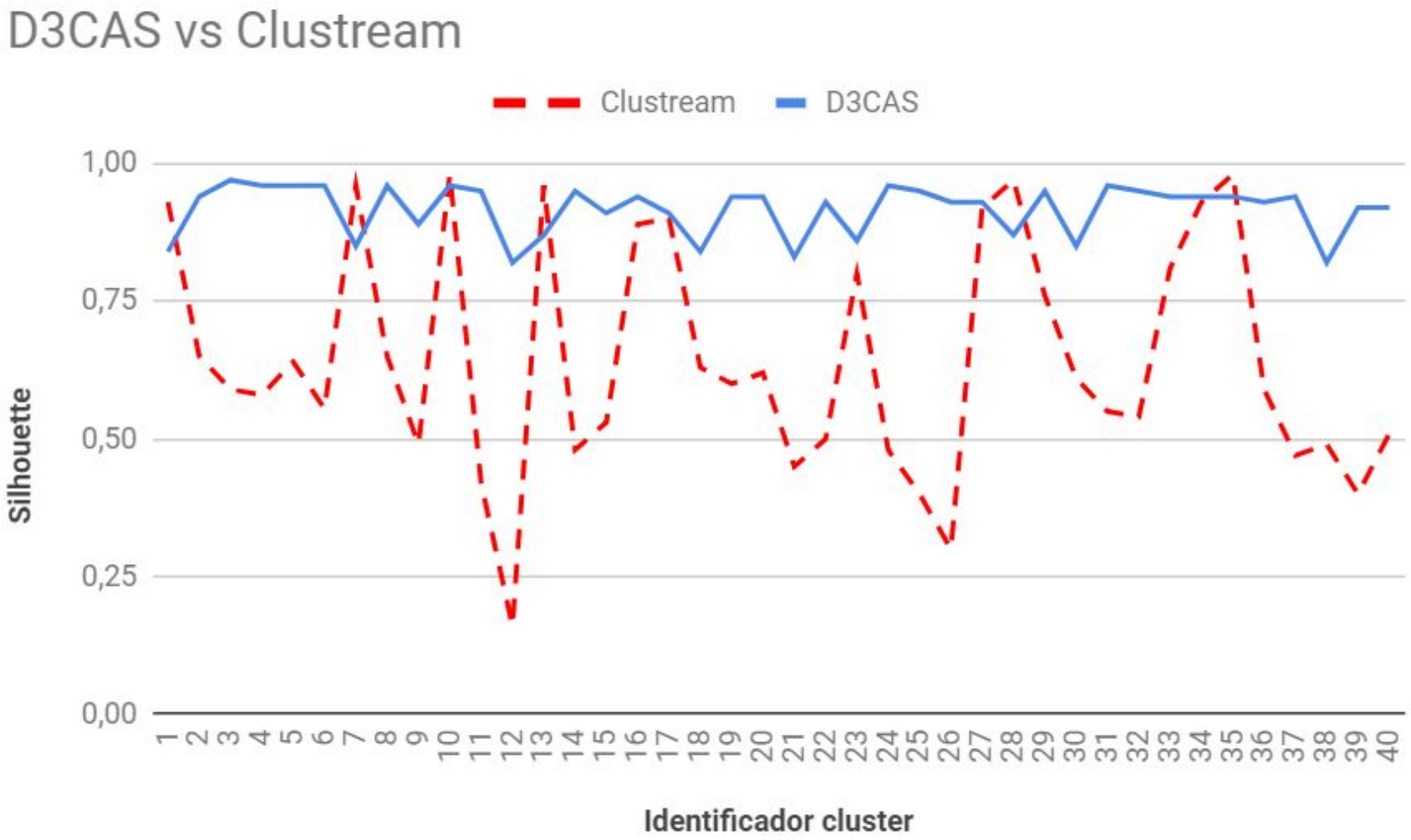
Izquierda: Clustream - Derecha: D3CAS



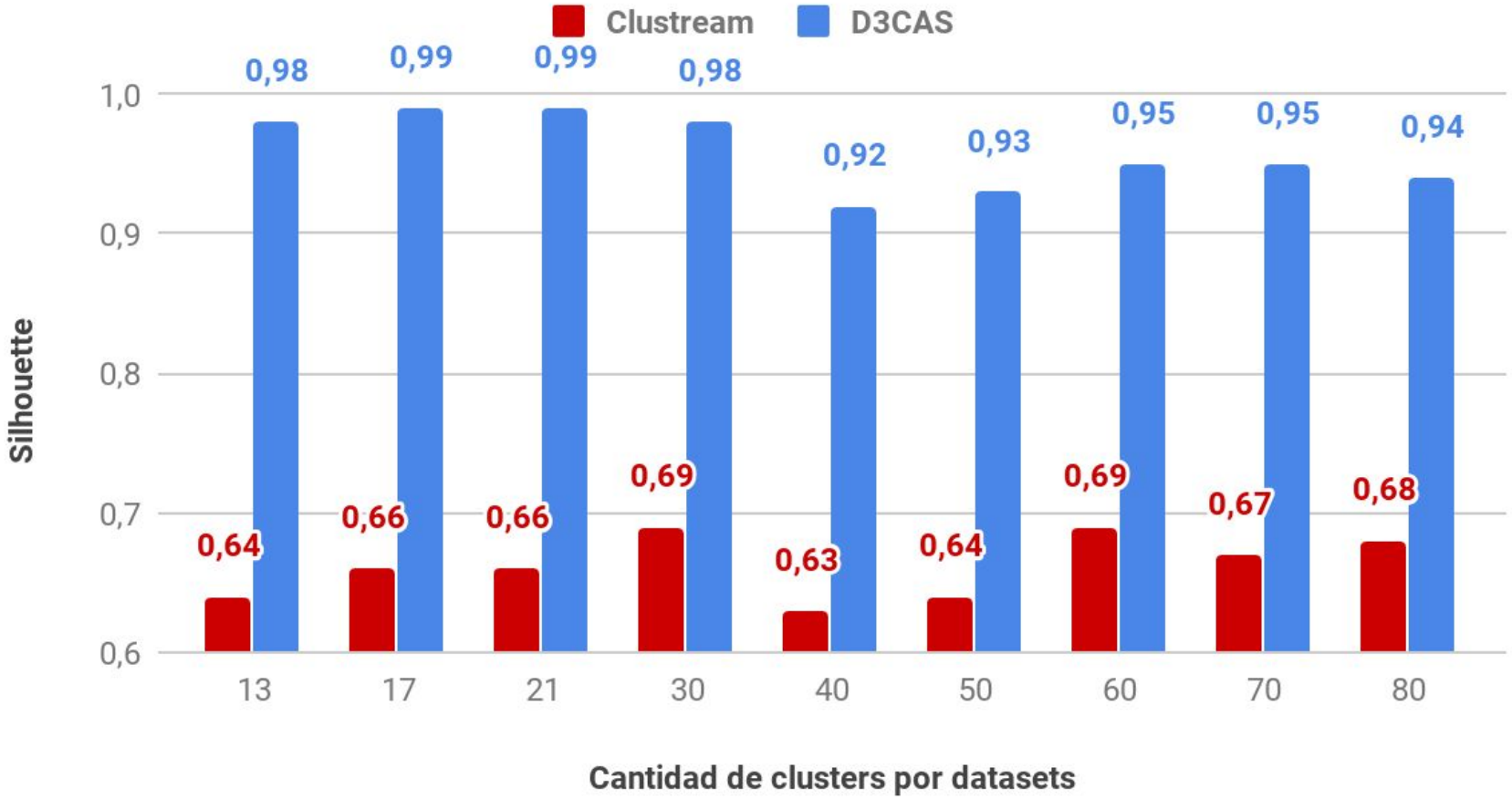
Silhouette 0,63

Silhouette 0,92

Dataset con 40 clusters

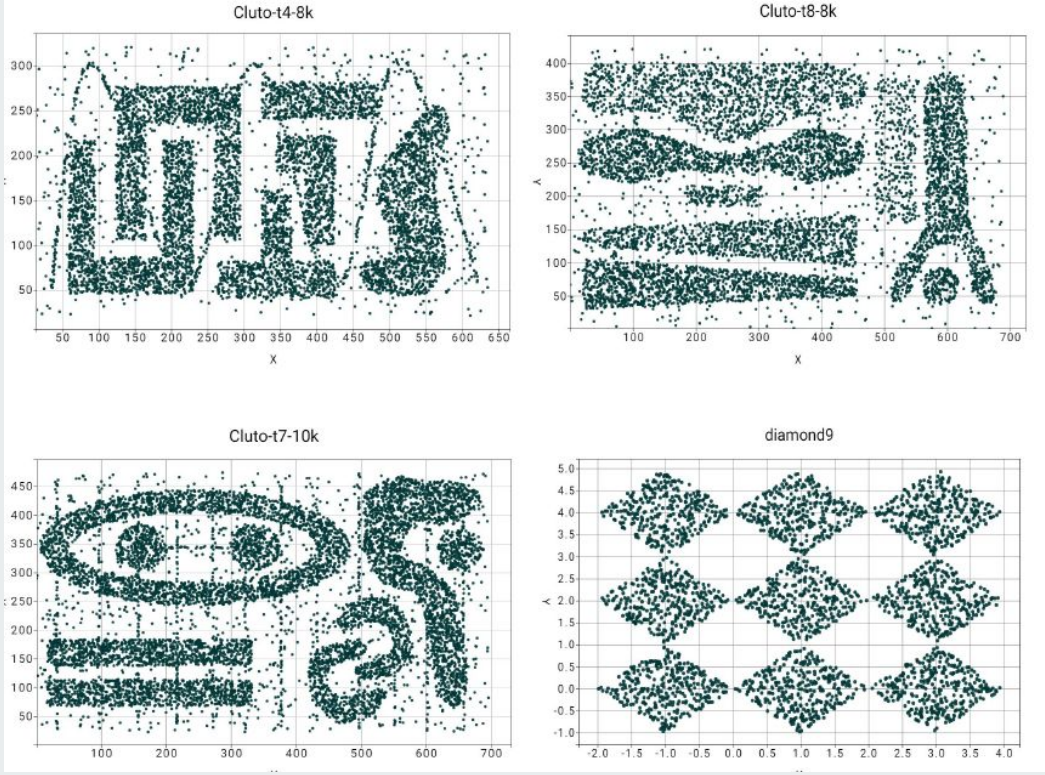


Coeficiente Silhouette

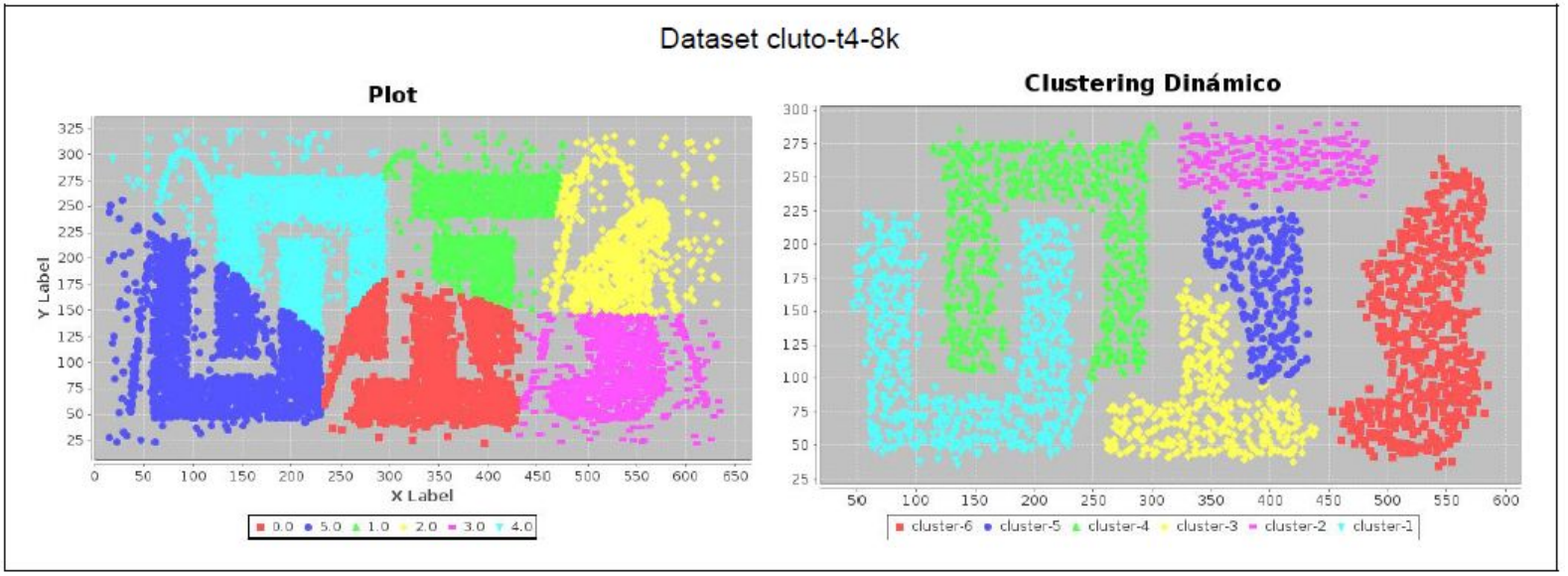


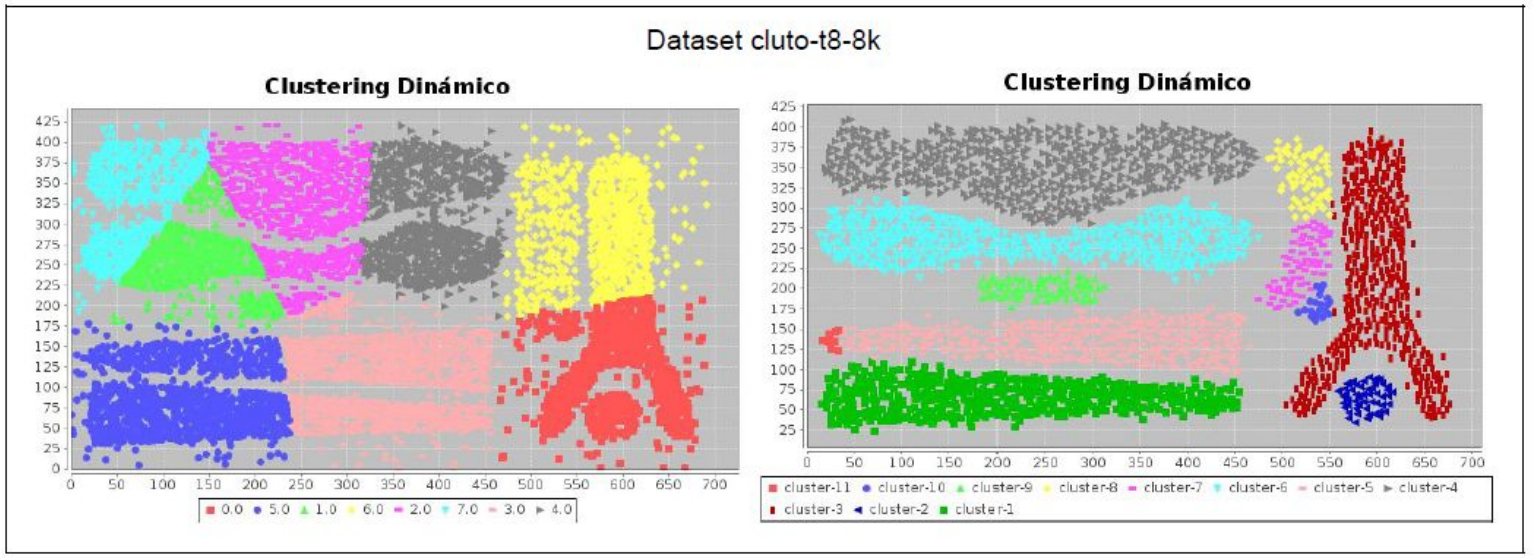
Detección de formas arbitrarias.

Criterio externo: **Pureza**.
Dataset Cluto: 10.000 datos.
Velocidad del Flujo: 1.000

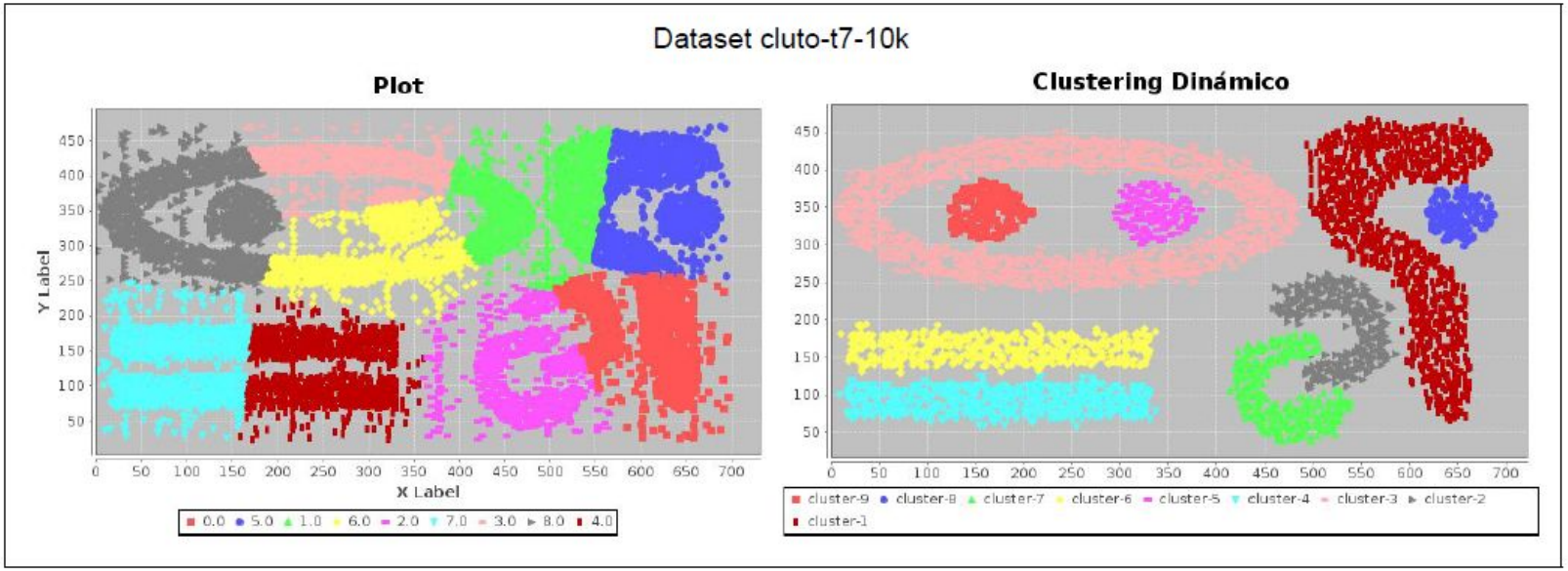


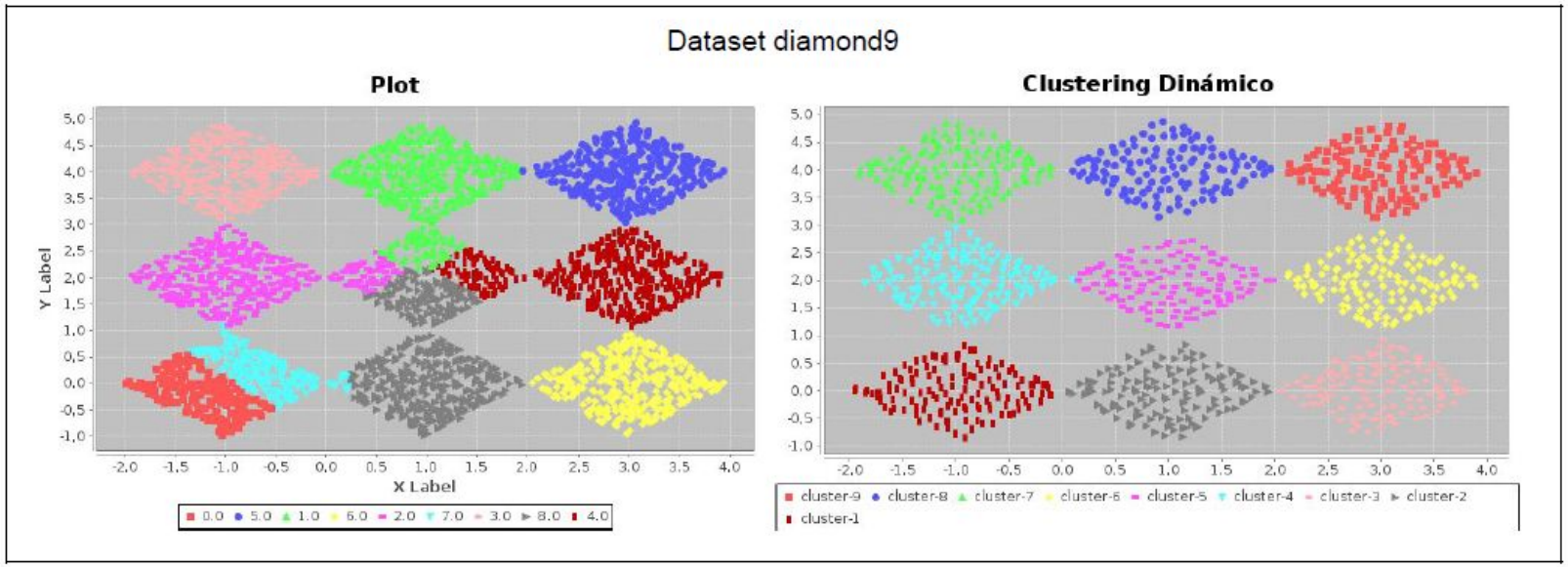
Izquierda: Clustream - Derecha: D3CAS



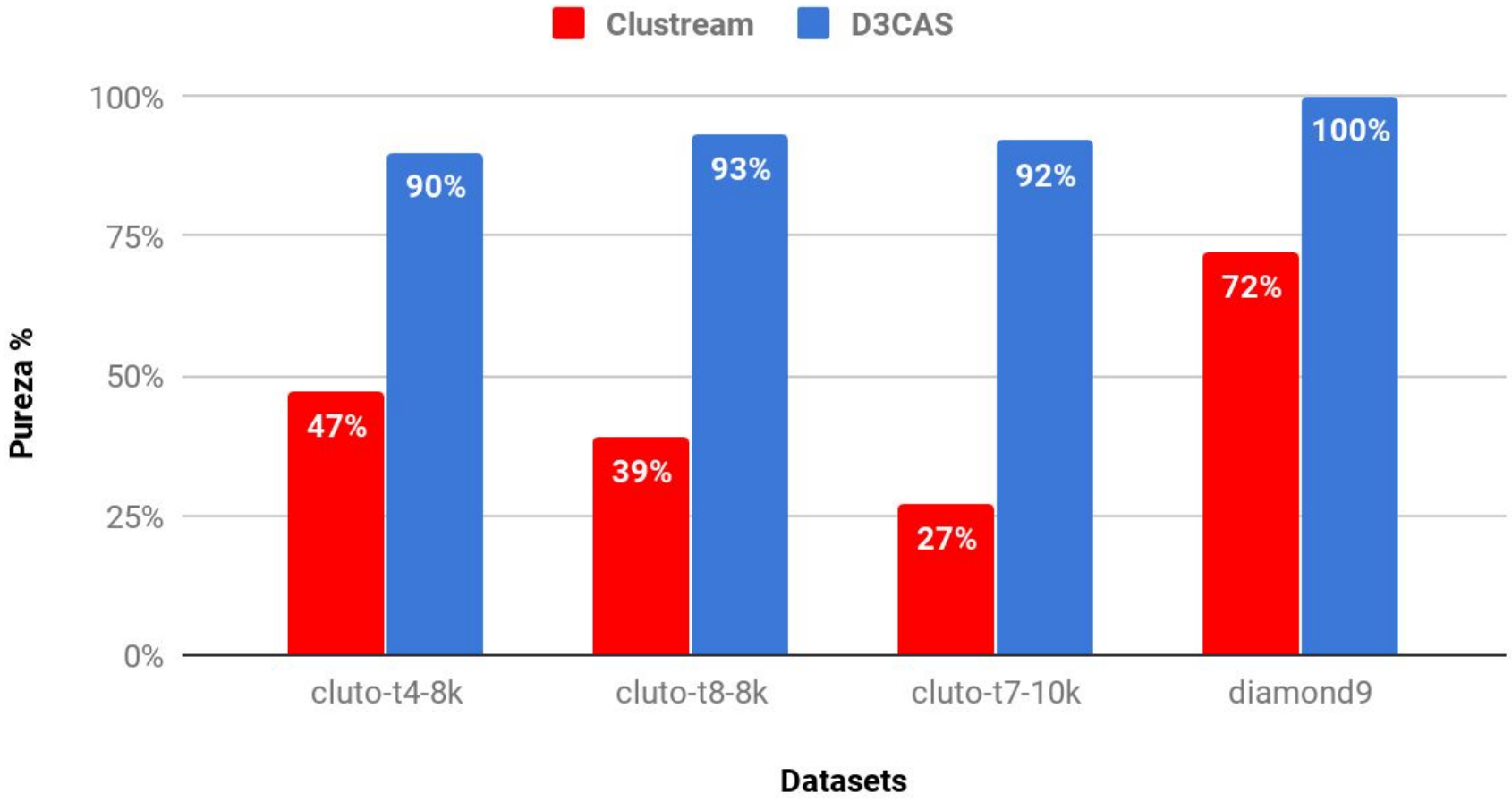


Izquierda: Clustream - Derecha: D3CAS



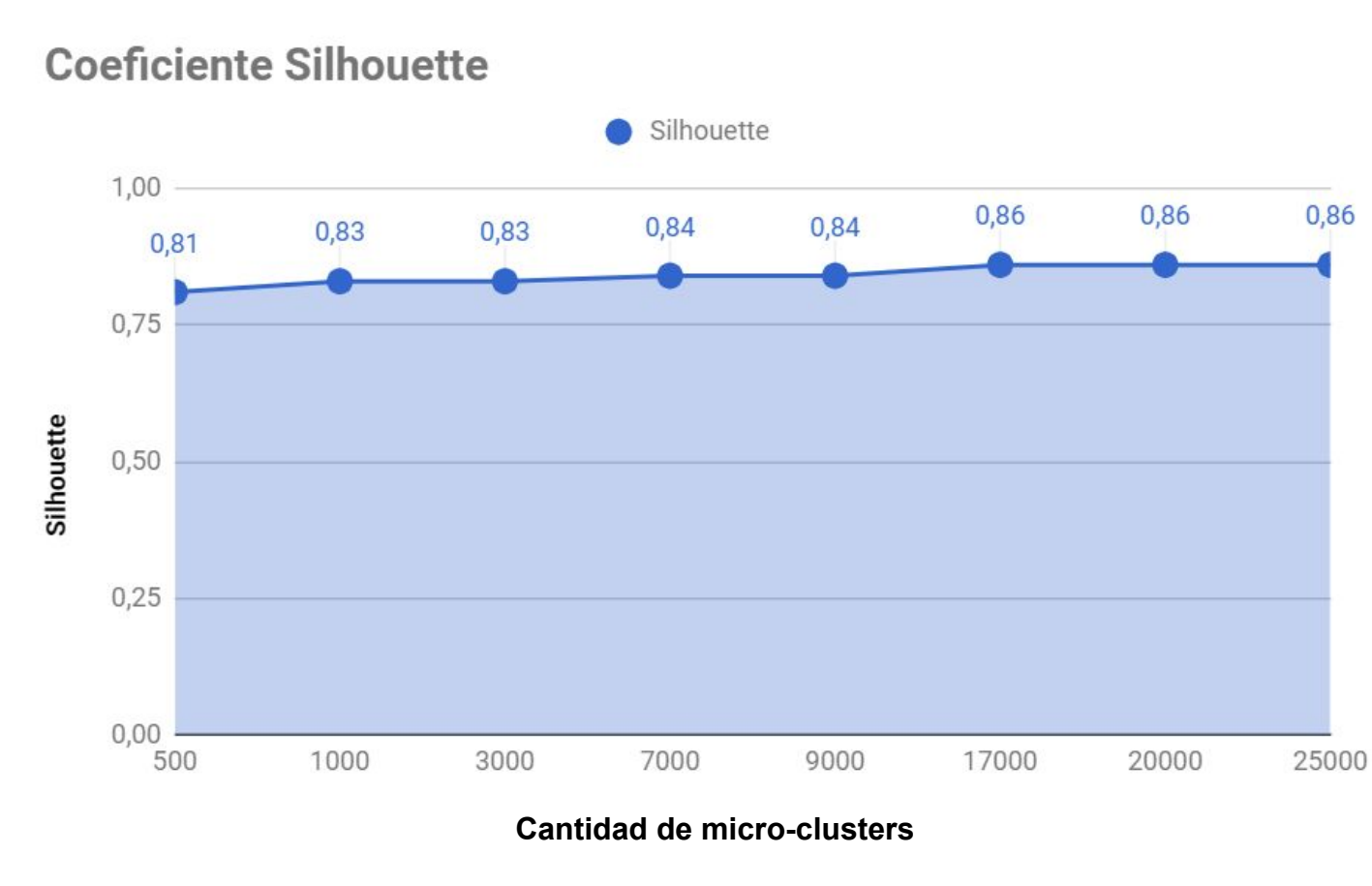


Metrica Externa: Pureza



Reducción del flujo.

Flujo de datos de 100.000 elementos.



Conclusión.

- Prueba de concepto de clustering sobre flujos de datos.
- Los experimentos realizados demuestran que con esta técnica se logra **buenos resultados**.
- Se logró implementar una técnica de clustering que trabaja sobre una **arquitectura distribuidas y escalable**.
- Por ser una técnica **dinámica de clustering** y estar basada en densidad, se la puede considerar como la **primer técnica** de clustering dinámica sobre **Spark**, ya que no hay trabajos similar publicados.
- Redacción de paper para **CACIC** 2018.

Trabajos futuros.

- Configurar un entorno real de cluster y realizar experimentos de D3CAS consumiendo un flujo de datos real como el que brinda el servicio de Twitter.
- Se propone continuar con la investigación de Spark, sobre todo, en la nueva versión (Spark Structured Streaming) a fin de utilizar las nuevas prestaciones de este motor y comparar el rendimiento entre las distintas versiones.
- Siguiendo la línea de D3CAS se propone un desafío mayor, como el de lograr implementar una técnica de clustering sobre el modelo de programación de GPU.

¿ Preguntas ?