TECHNICAL REPORT PERFILADO GARLANET

AUTOR: HAIZEA GÓMEZ RUIZ
DIRECTOR: JOAN MANUEL MARQUÈS PUIG
DEPARTAMENTO

CONSULTOR: ALEXANDRE VIEJO GALICIA

ABSTRACT

El estudio surge de la necesidad de perfilar la estabilidad y uptime de los nodos en un sistema de microblogging distribuido Volunteer Computing Grid. Para simular nuestros datos se han obtenido de la aplicación Seti@home, un sistema BOINC donde publica de forma masiva la información correspondiente a estas aplicaciones.

Inicialmente se realizó una taxonomía de aplicaciones similares para identificar si ya existía una tendencia en la gestión de logs para aplicaciones de microblogging.

Se establece un método de prefiltrado de la información (gestionando los logs de inicio y fin) y generando un vector de bits de actividad para la comparación de los nodos.

En el estudio se analizan distintas tipologías de clustering y sub-familias, para finalmente verificar con tres de ellas que cumplían los requisitos (KMeans basada en particionamiento, DB-SCAN basada en densidad y un metaheurístico propio).

Los resultados, obtenidos resultan adecuados con tiempos de computación, para K-Means siempre que se utilice la matriz de vectores de uptime como features, y utilizando la PCA (Principal Component Analysis) para reducir la dimensión del vector de bits a 2 dimensiones, se ha comprobado que en esta situación el perfilado es adecuado.

Concluyendo que tecnologías aplicadas actualmente al machine learning pueden ser válidas para el planteamiento al que nos hemos enfrentado, encontrándonos la limitación actual del modelo de almacenamiento de logs que proporcionan las aplicaciones.

PALABRAS CLAVE:

CLUSTERING, VOLUNTEER COMPUTING GRID, K-MEANS, LOGS, DB-SCAN.

INTRODUCCIÓN

Actualmente, la minería de datos se ha convertido en una necesidad para determinar los datos que manejan nuestras aplicaciones.
En términos generales, podemos decir que la minería de datos es el proceso de extraer información útil, patrones y tendencias previamente desconocidas, de grandes bases de datos.
La minería de datos ha recibido un gran impulso en los últimos tiempos motivado por distintas causas:

a) el desarrollo de algoritmos eficientes y robustos para el procesamiento de grandes volúmenes de datos,
b) un poder computacional más barato que permite utilizar métodos analizar patrones funcionales de las aplicaciones y/o de los clientes.
c) las ventajas comerciales y científicas que han brindado este tipo de técnicas en las más diversas áreas.

Clustering es una de las tareas más utilizadas en el proceso de minería de datos para descubrir grupos e identificar interesantes distribuciones y patrones en los datos.

En nuestro caso se pretende utilizar estas técnicas para dar una alternativa a la estabilidad de los servicios en soluciones basadas en Volunteer Computing Grid).

Garlanet, es un sistema de microblogging distribuido desarrollado por la UOC cuya pretensión es mantener la independencia de los proveedores de servicios.
Se basa en que los nodos son aportados de forma voluntaria.
El principal propósito de esta infraestructura es no tener las limitaciones de los sistemas de microblogging actuales, que son potencialmente vulnerables a ser coaccionados a:

- nivel político por entidades, gobiernos, empresas, limitando la gestión de sus contenidos,
- nivel de modelos de servicios, costes,..

Actualmente el sistema asocia el mismo grado de fiabilidad a cada uno de los nodos donde corre el servicio. Provocando en muchos casos la indisponibilidad de los mensajes que se encuentran almacenados por estar albergados en servidores menos estables.

La estabilidad de otros sistemas de microblogging distribuidos se basan en la replicación de la información en cada uno de los nodos, o la centralización de la misma en los sistemas del proveedor de servicios asociados a la aplicación de microblogging. Estas soluciones que van en contra de la filosofía de la solución.

Por lo que se pretende definir un modelado de datos que sea capaz de dar solución a la limitación actual, permitiendo analizar la aleatoriedad de los nodos y generar perfiles de candidatos aptos basado en un uptime adecuado para definirlo como candidato para almacenar los mensajes.

MOTIVACIONES

El análisis aquí planteado puede ser extrapolado a cualquier otro problema donde se pretenda calcular el perfilado de actores basado en el tiempo actividad como pudiera ser:

- El tiempo de permanencia de una identidad en una aplicación,

COMPUTACIÓN APORTADA VOLUNTARIAMENTE

La computación aportada voluntariamente (Volunteer Computing Grid o VCG) , fue una iniciativa extendida por Seti@home, que junto con la universidad de Berkley desarrolló la plataforma de BOINC, para unificar y concentrar los ciclos de CPU, almacenamiento (distribuido) que aportaba la gente de forma voluntaria. Donando parte de la potencia del ordenador o dispositivo para contribuir a investigaciones científicas que requieren de un super-computador o un grid para procesar la información

Esto se basa en instalarse un programa o agente que realiza esta función. El como y cuando se donan los recursos, depende del usuario y hacen que los nodos no sean igualmente fiables y sigan patrones distintos.

ESTADO DEL ARTE

En un primer momento, se realizó un análisis de aplicaciones similares para identificar como resolvían este problema, viendo que:

- bien utilizaban almacenamientos centralizados.
- O bien resolvían el problema, a través de la replicación del almacenamiento en cada nodo.

Este análisis queda fuera del ámbito de este estudio.

Por lo cual, en este análisis nos vamos a centrar en resolver el problema, a través de perfilar los nodos identificando aquellos que se comportan de una forma más estable, limitando el número de replicas a realizar.

Dentro de las soluciones de clustering, estas se resumen en supervisado y no supervisado, ya que el particionamiento está muy relacionado con la predicción de los siguientes valores o Machine Learning.
El clutering no supervisado está orientado a la categorización y resumir los datos. Únicamente requiere instancias (elementos a analizar con sus características).
A continuación se distinguen en una breve taxonomía de algoritmos de perfilado [7],[8].

SISTEMAS DE PERFILADO O CLUSTERS

Definición:

El clustering o perfilado de datos ayuda a discernir la estructura y simplifica la complejidad de cantidades masivas de datos. Es una técnica común y se utiliza en diversos campos como, aprendizaje de máquina, minería de datos, reconocimiento de patrones, análisis de imágenes y bioinformática [5], donde la distribución de la información puede ser de cualquier tamaño y forma.
La eficiencia de los algoritmos de clustering es extremadamente necesaria cuando se trabaja con enormes bases de datos y tipos de datos de grandes dimensiones.
En [8] se realizó un análisis de los procedimientos de perfilado, que vamos a seguir.

- **Elección de las variables.**
    - *Variables cualitativas:*
        - Ordinales (ejemplo nivel de estudios).
        - Nominales: (ejemplo nacionalidad).
    - *Variables cuantitativas*:
        - Variables discretas: (ejemplo nº de hijos)
        - Variables continuas: (ejemplo peso).
- **Elección de la medida de asociación.**
  Aquí hemos de distinguir si lo que se pretende agrupar elementos dentro de un cluster, para lo cual se ha de realizar un análisis de los elementos. O si se pretende agrupar “variables” más parecidas, para lo cual se ha de hacer un análisis del cluster de las variables, para ello basta en considerar la matriz de datos inicial X transpuesta (variando filas por columnas).

  Las medidas de asociación para elementos o distancias[1]:
    - Distancia de Minkowski (invariante ante translaciones)

$$dq(x_i, x_j) = \left\| x_i - x_j \right\|_q = \left(\sum_{l=1}^{p} \left| x_{il} - x_{jl} \right|^q \right)^{1/q}; q \geq 1$$

    - Distancia d1 o de ciudad (City Block) (con q=1):

[1] Estas distancias serán referenciadas en distintas partes del texto.

$$dq(x_i, x_j) = \|x_i - x_j\| = \left(\sum_{l=1}^{p} |x_{il} - x_{jl}| \right)$$

- Distancia euclídea (con q=2)

$$d_2(x_i - x_j) = \|x_i - x_j\| = \sqrt{\sum_{n=1}^{p} (x_i n - x_j n)^2}$$

- Distancia de Tchebychev o del máximo (con q=infinito):

$$d_\infty e(xi, xj) = \max\{l = 1, ..., p\} \mid x_l i - x_{lj} \mid$$

- Distancia de Mahalanobis

$$D_s(x_i, x_j) = \sqrt{(x_i - x_j)' S^{-1} (x_i - x_j)}$$

$\nabla xi \in Ci$

- **Elección de la técnica de clusters.**
  Definición de particionado (jerárquico (aglomerativos, disociativos), por partición, por búsqueda de densidad), que explicaremos detenidamente en un apartado.
  Existen también los métodos reductivos como el análisis factorial de tipo Q, o los métodos directos como el Block Cluster.
- **Validación de los resultados.**
  Definiremos un apartado a este punto.

En nuestro caso de análisis, tenemos un conjunto de variables cuantitativas continuas, donde nuestro perfilado pretende agrupar elementos (nodos de Seti@home) en clusters.

La elección del método de particionado y el método del cálculo de las distancias será definido una vez que se analicen los algoritmos de perfilado. De cara a la selección de la metodología hemos de tener en cuenta el impacto estimado de tiempo de computación en base a su eficiencia (**Error!**

Reference source not found.), y el tamaño de la muestra que vamos a analizar (véase **Error! Reference source not found.**).

Tabla 1 Impacto de tiempos en base a la eficiencia

n	10	100	1.000	10.000	100.000
O(n)	10 ms	0.1s	1s	10 s	100 s
O (n*log n)	33 ms	0.7 s	10 s	2 min	28 min
O (n^2)	100 ms	10 s	17 min	28 h	115 d
O (n^3)	1sg	17 min	12 d	31 años	32 milenios

ALGORITMOS DE PERFILADO

Describimos las familias de algoritmos de perfilado de conglomerados:

CLUSTERS POR PARTICIONES

Se denominan algoritmos basados en disimilaridad.

Características: Se debe fijar una “k” o número de clusters.

Algoritmos más representativos:

- k-Means (descrita en el procedimiento de análisis K-MEANS) [2]
- k-Modes (de 1998)

  Es una variante para utilizar atributos categóricos que implementa la modas en vez de medias aritméticas para el cálculo de los centroides. Esto es debido a que no se puede calcular la media (por no ser datos aritméticos) y se escoge el elemento central y frecuencias de aparición de elementos.
- k-Medoids

  Es una variante utiliza medianas en el método cálculo de la distancia de los centroides para solventar el problema de outliers.
  - PAM [Partitioning Around Medoids] (de 1987)
  - CLARA [Clustering for LARge Applications] (de1990)
  - CLARANS [CLAra based on RANdomized Search] (de1994)
- BCFR [Bradley, Fayyad & Reina]

  Es una variante para conjunto de datos muy grandes, en este cálculo se basa en calcular la probabilidad de que un nodo se encuentre en un cluster, por la distancia al centroide, dimensión a dimensión. Para ello utilizamos la distancia de Mahalabonis.
- K-prototipos: Es una alternativa en donde tenemos datos mixtos, donde por una lado se utilizará la distancia euclídea para los datos numérico y el segundo (no numérico) se usa la moda. A este segundo término se asigna un peso para poder ponderarlo y disponer de un resultado numérico.

$$d2(X,Y) = \sum_{1}^{m}(\sum_{j}^{p}(xi - yi)^2 + \gamma \sum_{j=i+1}^{m} \delta(xj, yj))$$

Amplía la posibilidad de usar K-Modas para datos categóricos, y utilizar una solución similar a K-Means.

K-MEANS

Posiblemente sea el algoritmo de particiones más conocido y testeado debido a que comparativamente a aportado los mejores rendimientos basados en pruebas.
Es un método que es sensible a la presencia de outliers, por lo que se precisa un pre-filltrado de estos elementos. Su uso no es adecuado cuando los clusters son no convexos, de distinto tamaño o de distinta densidad.

Este método se basa en hacer perfilados conociendo el número de clusters (k).

Cada cluster tiene asociado un centroide (centro geométrico del cluster).
Los puntos se asignan al cluster cuyo centroide esté más cerca (utilizando cualquier métrica de distancia).
Iterativamente, se van actualizando los centroides en función de las asignaciones de puntos a clusters, hasta que los centroides dejen de cambiar.

Limitaciones:
Existen limitaciones adicionales para la implementación de esta solución:

- Cluster no convexos, de distinto tamaño o de distinta densidad.
- Selección adecuada de los centroides.
- Selección del algoritmo para el cálculo de centroides.
- Atributos a comparar no son numéricos.

Soluciones:

Solución para clusters no convexos, de distinto tamaño o de distinta densidad:

- Se implementa K-Means con valores de “k” elevados y se revisan los resultados obtenidos.
- Se implementan soluciones alternativas como “Spectral clustering”

Soluciones para la selección de centroides:

1 – Partir de centroides escogidos al azar, pero realizar las suficientes iteraciones de medición de la distancia entre nodos, seleccionando los nuevos centroides y comparar los resultados obtenidos (GRASP) .

2 – Escoger centroides de partida potenciales (pre-calculados), para ello existen distintos métodos (k-means++).

3- Seleccionar un método jerárquico para determinar el valor de “k” y posteriormente implementar un sistema basado en particiones.

Solución para comparar atributos que no son numéricos:

Nota: Aunque K-Means es un algoritmo válido para strings (usando la distancia de Levenshteing) que es similar a la distancia de Hamming (binaria), aquí nos referimos a datos no estructurados.

Nota: La similaridad entre dos cadenas de texto A y B se basa en el conjunto mínimo de operaciones de edición necesarias para transformar “A” en “B”, o viceversa. Hay tres operaciones de edición, las cuales son destrucción, inserción y substitución. Entre más cerca de cero es la distancia de Levenshtein más parecidas son las hileras:

`Pseudocódigo`:

1- Fijar un patrón numérico que mantenga una equivalencia entre un valor de un atributo y un valor numérico o binario, que facilite su cálculo.

D(elemento1, elemento2) = 1 si elemento1 == elemento2 :0 en cualquier otro caso.

En este caso, el método para calcular la distancia entre componentes para el calculo del centroide sería la moda, variante de K-Means, denominado K-Mode.

Método de cálculo:

Inicialización :

- Escoger k centroides aleatoriamente (o escoger nodos al azar).
- Formar k grupos, asignando cada punto al centroide más cercano.
-

Proceso iterativo

- Mientras que los centroides cambien:
- Calcular las distancias de todos los puntos a los k centroides. Formar k grupos, asignando cada punto al centroide más cercano.
- Recalcular los nuevos centroides.

Nota: La forma de recalcular los centroides:

$$c_j = \frac{1}{|C_j|} * \sum_{\forall x \in C_j} z | z \in C_j$$

donde "c_j" es el centroide y "z" es un elemento que pertenece al cluster "C_j".

Se pretende minimizar las distancias inter-cluster o separación entre clusters e intra-cluster o cohesión de los nodos.

Algunas definiciones con respecto a las distancias:

La distancia promedio inter-clústeres es la distancia promedio entre los centroides de los clusters. Véase dibujo

• La distancia mínima inter-clusters (centroides) es la distancia mínima entre dos centroides de dos clusters.

• La distancia mínima entre clusters (nodos) es la distancia mínima entre un nodo y el centroide de un clúster al que no pertenece.

• La distancia promedio dentro del clúster (intra-cluster(es la distancia promedio entre un nodo y el centroide del clúster al que pertenece.

• La distancia máxima intra-clúster es la distancia máxima entre un nodo y el centroide del clúster al que pertenece.

Complejidad Tiempo: $O(n * k * I * d)$

Donde n = número de puntos, k = número de clusters, I = número iteraciones, d = número de atributos

Complejidad en el espacio: $O((N + K) * D)$

Comparativa de eficiencia entre métodos:

PAM, limitado a su complejidad computacional (escalabilidad), es válido para conjunto de datos pequeños.

Comparativa de eficiencia:

PAM $O(k * (n - k)^2)$ vs K-Means $O(n * k)$

MÉTODOS BASADOS EN EL ANALISIS DE DENSIDAD

Este tipo de análisis es efectivo cuando su forma es irregular, están entrelazados o existe ruido y outliers de datos.

Características:

- Se identifican a los clusters de forma arbitraria.
- Robustos ante la presencia de ruido (puntos dispersos que no forman parte de un cluster ni están en su frontera) .
- Escalables: Un único recorrido del conjunto de datos.

Algoritmos más comunes:

- DBSCAN: Density Based Spatial Cluster Applications with Noise (KDD 1996)
- OPTICS: Ordering Points To Identifiy The Cluster Structure (Ankerst et al SIGMOD 1999)
- DENCURE: Density based Clustering (Hinnedburg & Keim, KDD 1998).

DBSCAN:

Es el más extendido de los clusters de densidad por su eficiencia y facilidad ya que requiere pocos parámetros de entrada, para establecer una partición:

Elimina el ruido y agrupa los datos restantes:

Se han de determinar los parámetros :

- Epsilon (radio del vecindario o del cluster).
- minPts (mínimo número de puntos del vecindario o cluster).

Eficiencia tiempo: O (n * tiempo necesario para encontrar vecinos en la distancia Epsilon)

En el peor caso: $O\ (n^2)$

En el mejor caso: O(n * log n).

Eficiencia en espacio: $O\ (n^2)$

Limitaciones:

No es capaz de detectar clusters anidados.

Solución:

Implementar la alternativa de **enDBSCAN** (modificación de DBSCAN capaz de detectar cluster anidados en espacios de densidad variable).

Pseudocódigo:

```
current_cluster_label <- 1
for all core points do
      if the core point no tiene cluster_label  then
            current_cluster_label=current_cluster_label +1
            Label the current core point with the cluster
label current_cluster_label.
      endif
  for all points in the Eps-neighborhood except position
i the point ifself do
      if the point does no have a cluster label then
            Label the point with the cluster label
current_cluster_label
      End if
  end for
end for
```

Se agrupan en base a su densidad y el radio prefijado.

CLUSTERING JERÁRQUICOS

Características: No se fija la k".

Algoritmos más representativos:

- BIRCH: Balanced Iterative Reducing and Clustering using Hierarchies (Zhang, Ramakrishnan & Livny, SIGMOD'1996)
- CURE: Clustering Using Representatives (Guha, Rastogi & Shim, SIGMOD'1998)
- ROCK: Robust Clustering using links (Guha, Rastogi & Shim, ICDE'1999)
- CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (Karypis, Han & Kumar, 1999

Segundo método de agrupamiento que podría utilizarse para solventar los problemas de la preselección de "k" número de clusters y no necesitar eliminar los outliers sería utilizar los métodos jerárquicos. Además no están limitados por la comparación de clusters de distintas densidades.

Una vez que se conocen las distancias existentes entre cada dos elementos se observa cuáles son los individuos más próximos en cuanto a esta distancia o similaridad. Estos formarán un grupo que no vuelve a separarse durante el proceso. Se mide la distancia o similaridad entre todos los elementos de nuevo (tomando el grupo como un elemento) de la siguiente forma:

- Cuando se mide la distancia entre el grupo formado y un elemento, se toma la distancia mínima de los elementos del grupo al nuevo elemento,

- Cuando se mide la similitud o similaridad entre el grupo formado y un elemento, se toma la máxima de los elementos del grupo al nuevo elemento.

Distancia	A	B	C	D
A	0			
B	9	0		
C	4	5		
D	7	3	11	0

Tabla 2 Ejemplo de matriz de similitudes

Distancia mínima:

Entre B y D (3) -> B y D forman un grupo, valor de B-D(7) máximo que da D.

Se miden las distancias de nuevo:

Nueva distancia mínima: C y A(4) -> A-C conforman un grupo (valor 5).

Se unen los grupos A-C y B-D formando el grupo A-B-C-D.

Aunque matemáticamente tienen una complejidad mayor y se desaconsejan para grandes volúmenes de datos se han utilizado de forma correcta en análisis como perfilado en el análisis de datos orientado a la diabetes (gran volumen de información), análisis neuronales,….

Estos métodos de forma gráfica se puede representar en forma de dendogramas o diagramas de datos basados en árboles. Agrupando los objetos por su representante y su similitud a través de la altura.

El dendograma es un gráfico usado que permite visualizar el procedimiento de agrupamiento del cluster por pasos, y que ayuda a decidir el número de grupos que podría representar la mejor estructura de datos teniendo en cuenta la forma en que se va generando el cluster y la media de similitud y las particiones que se van generando.

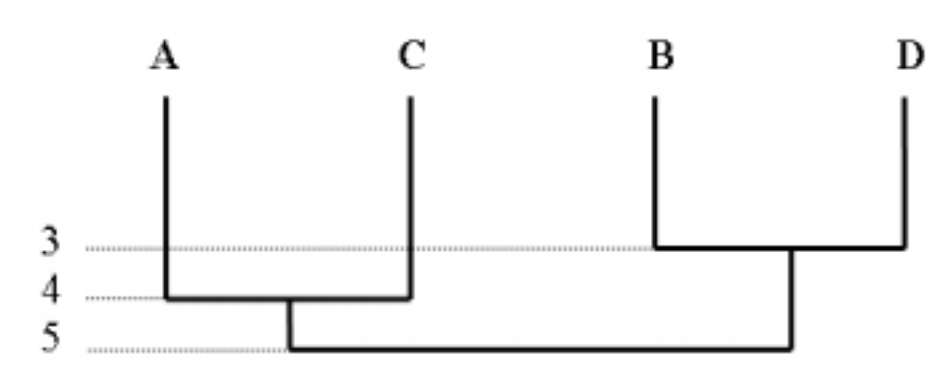


Figure 1 Dendograma

Aquí partimos que lo que nos interesa es saber el número de clusters (valor "k" de K-Means) ´óptimo.

Siempre que utilicemos una muestra baja, ya que el coste computacional es alto.

Este método nos identifica los outliers de forma que no se requiere pre-procesar los datos.

Aquí para agrupar los datos podemos seguir dos tendencias de clasificación:

- Técnicas Divisivas (**DIANA**: DIvisive ANAlisys), partiendo de un conjunto de datos (todos los datos) vamos de forma recursiva generando subconjuntos de datos cohesionados llegando a un cluster o a "n" clusters.

- Técnicas Aglomerativas (**AGNES**: AGlomerative NESting), de un subconjunto de datos perfilados "clusters" componemos el árbol de relaciones.

Su escalabilidad es del orden:

Complejidad en el espacio: $O > (n^2)$,

Complejidad en el tiempo: $O\ (n^2 * \log n)$

para solventar este problema, se escogen variantes que vamos a describir:

BRICH se basa en un árbol balanceado (cuyos nodos almacenan las sumas

de los Clusters Features (CF) de sus descendientes.

Es optimo para grandes tamaños porque su escalabilidad es lineal O (n).

Limitaciones:

- Sólo es válido para datos numéricos.
- Se debe fijar a priori el tamaño de clusters (para poder dividirlo en ramas) y el cluster pueden resultar no naturales

Generan una estructura de datos jerárquica CF Tree,

Los CFs tienen dos parámetros, el primero el número máximo de hijos de sus sub-nodos y el segundo el diámetro máximo de los sub-clusters. Con ambos parámetros determinados la densidad y tamaño del cluster y el peso que tiene el mismo.

Se debe determinar el número máximo del diámetro del cluster para generar una agrupación adecuada.

El procedimiento se basa en:

Para cada nuevo punto, y ver cual es la hoja más cercana, ***entonces*** se añade a su cluster y actualizar el CF.

Si, el nuevo CF tiene un diámetro mayor al admitido, **entonces**

se sub-divide el cluster en dos y se actualizan los CFs

De esta forma establecemos nuestra estructura arborescente.

La variante más utilizada en las investigaciones para los métodos jerárquicos es **CURE** (Clustering Uses Represtantive) al que no le afectan clusters con formas convexas.

Al igual que K-Means utiliza un centroide calculado con la distancia euclídea[2] entre puntos, al cual le añade la información de puntos representativos, para poder ser inmune a la forma del cluster. Estos valores dispersos que serían outliers, se les acerca al centroide una distancia marcada por % definido para que puedan incluirse en el cluster en vez de prescindir de ellos.

La complejidad del algoritmo es para un espacio de orden O (n), tiene un tiempo de computación del orden O (n^2 * log n) .

Esto nos limita el número de elementos a procesar. Esto es debido que este tipo de clusters se implementan en memoria principal.

El algoritmo se basa en dos etapas:

1º Etapa:

- Seleccionar una muestra de datos y almacenarla en memoria.
- Agrupar los puntos siguiendo un algoritmo jerárquico.
- Seleccionar los representantes y acercarlos al cluster.

2º Etapa:

- Recoger el conjunto completo de datos y asociar a cada punto su cluster más cercano.

A nivel de pseudocódigo,

```
For every cluster u (each input point),
      u.mean = mean of the points in the cluster
      u.rep = set of c representative points of the cluster
      (initially c = 1 since each cluster has one data point).
      u.closest = the cluster closest to u.
All the input points are inserted into a k-d tree T
```

[2] Distancia euclídea: $D_{ij}=\sqrt{(\sum_{k=1}^{n}(x_{ki} - x_{kj})^2)}$

```
Treat each input point as separate cluster, compute u.closest
for each u and then insert each cluster into a heap Q
(clusters are arranged in increasing order of distances
between u and u.closest)

while size(Q) > k
      Remove the top element of Q (say u) and merge it with its
      closest cluster u.closest (say v) and compute the new
      representative points for the merged cluster w.
      Remove u and v from T and Q. For all the clusters x in Q,
      update x.closest and relocate x
      insert w into Q
```

ASPECTOS GENERALES

A efectos de medir la calidad de los clusters encontrados por los distintos algoritmos anteriormente descritos, hemos considerado las medidas intra-cluster e inter-cluster, que nos determinan la cohesión y la separación entre cluster respectivamente.
Vamos a describir las formulas que implementaremos :

Cuantificación del error:

$$\mathbf{1\ Error-cluster} = \frac{\sum_{i=1}^{k}(\sum d\frac{d(ci-Z)}{|Ci|})}{K}$$

Distancia Intra-Cluster:

$$\mathbf{2\ Intra-Cluster} = \frac{\sum_{i=1}^{K}(\sum_{z,t\in Ci}\frac{d(z,t)}{|Ci|})}{K}$$

Distancia inter-Cluster:

$$\mathbf{3\ Inter-cluster} = \frac{\sum_{i=1}^{K-1}\sum_{j=i+1}^{K} d(ci,cj)}{\sum_{i=1}^{K-1} i}$$

OTROS ALGORITMOS:

Con el fin de que no parezcan que las soluciones únicamente se basan en las anteriormente indicadas. [4]
Por mencionar algunos adicionales:

- Clúster basados en el vecino más próximo. (k-medoids).
- Métodos basados en rejillas.
- Clústeres probabilísticos
- Métodos basados en Co-ocurrencia de datos categóricos.
- Clúster basados en restricciones.
- Clúster sub-espacial.

CLUSTER PROBABILISTICO

Vamos a hacer una referencia como alternativa a nuestra solución.
Existen algoritmos basados en modelos, en estos los datos son generados por modelos probabilísticos(cadenas de Markov o variantes de estas) y estimar sus parámetros. [8]
Esto le aporta una gestión de dinamismo a los análisis.
La idea de las propuestas de Cadez et al. (2003) y Pamminger and Frühwirth-Schnatter (2010), suponiendo el primero un modelo basado en cadenas de Markov para definir el comportamiento de los usuarios de la web www.msnbc.com a través de las distintas categorías de información que ofrece la página (actualidad, deportes, tiempo, viajes, página principal, ...).
Para ello, gracias a las cookies de cada usuario y por medio de una transformación de los datos obtenidos a través de ellas, se ha simplificado la información pasando a obtener una secuencia de entradas a categorías por cada uno de los visitantes de la web.

Lo importante es que se supone que cada cluster va seguir un modelo de Markov distinto. Y esto aporta el planteamiento consigue indicar que la heterogeneidad existente en los distintos clusters debida a la diversidad de la duración de la estancia en cada categoría, los contenidos visitados, . . . los datos tratados en los ejemplos del artículo son modelizados como si se hubiesen generado de la siguiente manera.

1) Un usuario llega a la red y se le asigna a un cluster con cierta probabilidad.
2) El comportamiento del usuario se genera a partir de un modelo de Markov con parámetros específicos de ese grupo.

Así pues, debe de estar concretado el comportamiento de los individuos para cada uno de los clusters. para determinar la proporción de usuarios existentes en cada cluster y especificar los parámetros de cada modelo de Markov, se utiliza el algoritmo EM.

EM

Pertenece a una familia de modelos que se conocen como Finite Mixture Models, los cuales se pueden utilizar para segmentar conjuntos de datos. Dentro de la clasificación, encajaría dentro de los Métodos de Particionado y Recolocación, en concreto Clustering Probabilístico. Se trata de obtener la FDP (Función de Densidad de Probabilidad) desconocida a la que pertenecen el conjunto completo de datos.

COBWEB

Se trata de un algoritmo de clustering jerárquico. COBWEB [4], se caracteriza porque utiliza aprendizaje incremental, esto es, realiza las pariciones instancia a instancia.
Al algoritmo COBWEB no hay que proporcionarle el número exacto de clusters que queremos, sino que en base a los parámetros anteriormente mencionados encuentra el número óptimo

Durante el proceso el algoritmo genera un árbol (árbol de clasificación) donde las hojas representan los segmentos y el nodo raíz engloba por completo el conjunto de datos de entrada.
Al principio, el árbol consiste en un único nodo raíz y las instancias se van añadiendo una a una al árbol, y este se va actualizando.
La actualización consiste en encontrar el mejor sitio donde incluir la nueva instancia, operación que puede necesitar de la reestructuración de todo el árbol (incluyendo la generación de un nuevo nodo anfitrión para la instancia y/o la fusión/partición de nodos existentes) o simplemente la inclusión de la instancia en un nodo que ya existía. La clave para saber cómo y dónde se debe actualizar el árbol la proporciona una medida denominada utilidad de categoría, que mide la calidad general de una partición de instancias en un segmento.
La reestructuración que mayor utilidad de categoría proporcione es la que se adopta en ese paso.
El algoritmo es muy sensible a otros dos parámetros:

- Acuity Este parámetro es muy necesario, ya que la utilidad de categoría se basa en una estimación de la media y la desviación estándar del valor de los atributos, pero cuando se estima la desviación estándar del valor de un atributo para un nodo en particular, el resultado es cero si dicho nodo sólo contiene una instancia. Así pues, el parámetro acuity representa la medida de error de un nodo con una sola instancia, es decir, establece la varianza mínima de un atributo.
- Cut-off Este valor se utiliza para evitar el crecimiento desmesurado del número de segmentos. Indica el grado de mejoría que se debe producir en la utilidad de categoría para que la instancia sea tenida en cuenta de manera individual. En otras palabras: cuando no es suficiente el incremento de la utilidad de categoría en el momento en el que se añade un nuevo nodo, ese nodo se corta, conteniendo la instancia otro nodo ya existente.

Véase posteriormente a las limitaciones y falsos positivos:

De cara a nuestro análisis se basa en usar un conjunto de parámetros “t” que nos identifican a nuestros clusters , sin requerir utilizar todos los valores.
Y presentando una complejidad de `O(k*n*t)` con k=número de clusters, y “n” número de elementos, mejorando los tiempos de ejecución de los modelos de particionamiento.

El planteamiento se basa en base a la %average se le asocia probabilísticamente a un cluster.
Una vez determinado el cluster, su comportamiento viene definido a través de un modelo estadístico (cadena de Markov) propio del cluster al que pertenece. Con estas suposiciones acerca de la estructura, los objetivos del algoritmo son calcular el número de clusters, hallar la función de probabilidad que asigna un vector de uptime a un cluster en particular y determinar los parámetros de cada cluster.

Suponiendo que esta composición (denominada estructura de modelos mixtos) consta de k conglomerados, se puede considerar que la probabilidad de que conocidos los parámetros θ del modelo, se obtenga el elemento X viene determinada por:

$$p(X|\theta) = \sum_{i=1}^{k} p(c_i|\theta)\, p_i(X|c_i, \theta)$$

- X = [x1,..., xm] representa un vector de bits. Así, X determina un elemento del conjunto de datos, con valores concretos xi para cada una de las categorías. Siendo i ∈ {1,..., m}.
- θ hace referencia al conjunto de parámetros que describen el modelo considerado. los cl : l ∈ {1,..., k} se consideran indicadores de cada uno de los k clúster supuestos para el modelo.
- p(cl | θ) es la probabilidad marginal de que un elemento pertenezca al “i – ésimo” cluster. Como consecuencia inmediata se tiene que $\sum_{i=1}^{k} p(c_i|\theta) = 1$

- pl(X | ci ,θ) es el modelo estadístico que describe la distribución para las variables dentro del “i – ésimo” cluster; es decir la probabilidad de obtener el vector de bist X sabiendo que pertenece al cluster especificado.

GESTIÓN DE LOGS

Dado que no se dispone de logs reales para el análisis se ha propuesto el uso de los logs que aporta Seti@home que son compatibles con la solución de Garlanet.
De cara al análisis de los datos, se pretende perfilar los nodos en base a su actividad.

En nuestros casos vamos a partir de un conjunto de una cantidad de información a procesar muy elevada que nos propone su gestión basada en el análisis de fichero, o basado en querys a una base de datos relacional (mysql).
El fichero de logs análisis de partida extraído de la página FTA (Failure Trace Archive, http://fta.scem.uws.edu.au/data/seti09/seti09_tab.tgz) nos proporciona un archivo comprimido en formatos tar.gz para su análisis comprendidos entre las fechas 2007-2009.

De entre los ficheros o tablas que nos propone la FTA, se selecciona el fichero event_trace.tab cortado , ya que el original es de 18Gb de información.
Este fichero dispone de la información relativa a los estados de actividad.

Tabla 3 Ejemplo de tabla de logs event_trace.tab

event_id	component_id	node_id	platform_id	node_name	event_type	event_start_time	event_stop_time	event_end_reason
0	0	100416828	2	100416828	1	1211524407.906	1211524417.922	NULL
1	0	100416828	2	100416828	0	1211524417.922	1211524440.312	NULL
2	0	100416828	2	100416828	1	1211524440.312	1211615915.156	NULL
3	0	100416828	2	100416828	0	1211615915.156	1211616057.031	NULL

Descripción de los campos:

- event_id: Id del evento.
- component_id: es una relación con la tabla de component.tab (donde se especifican los traces de los eventos para cada componente).

- node_id: es una relación con la tabla node.tab (siendo este elemento la PK de esa tabla y coincidente con el nombre del nodo).
- event_id desde 0, el event_start_tine coincide con su event_stop_time del otro.
- Platform_id: Indice la plataforma.tab, 2 identifica a la plataforma de Seti@home.
- Node_name: referencia a la tabla node.tab, donde esta referencia identifca al tipo de nodo (arquitectura, sistema operativo, número de cores, tamaño de disco aportado, memoria,..).
- Event_type: Campo que nos va a identificar si estamos activos o inactivos.
- Event_start_time: Hora de actividad en formato epoch.
- Event_stop_time: Hora de parada en formato epoch.

A modo de resumen:

- Del valor de event_trace,tab event_type (0|1) definen los estados de actividad.
- Del fichero event_trace el code 0 unanvailability, 1 availability, 2 user present y 3 CPU excedida, no aporta valores (que pudieran ser requeridos).
  Existe en el : http://fta.scem.uws.edu.au/index.php?n=Tools.Analysis hay un conjunto de herramientas que no vamos a utilizar, bien porque no aportan a nuestro análisis o porque hacen referencias a librerías ya obsoletas.

Dado el tamaño de nuestra muestra, se ha escogido una porción de esta información que comprende el periodo de tiempo de:

Inicio: 08:37:27 Viernes 23 de mayo de 2008.

Fin: 09:50:46, Domingo 28 de septiembre de 2008

Que corresponden a 10.000.000 de líneas de logs con un tamaño de 893MB.

De este fichero base de partida, se han extraído distintas muestras para realizar los ensayos con las siguientes características:

Tabla 4 Tamaños de las muestras disponibles

Líneas procesadas	Tamaño	Nº de nodos distintos
1.000	89 (KB)	4
10.000	889 (KB)	30
100.000	8,7 (MB)	118
1.000.000	87 (MB)	1.093
10.000.000	893 (MB)	11.236
202.546.161	18 (GB)	226.209

A través de esta información se puede deducir que el comportamiento de los nodos que continuamente se van a conectar y desconectar.
Esto se aprecia

METODOLOGÍA

En el procedimiento a realizar, pretendemos cotejar dos métodos de perfilado.
El primero de ellos, denominado perfilado grueso, a grandes rasgos se basa en reducir los logs de actividad de los nodos (una vez pre-procesados, filtrados) a un vector de bits, donde vamos a aplicar un método personalizado de filtrado.
Esta metodología, la queremos cotejar con el uso los métodos frecuentes de perfilado (particiones, densidad, jerárquicos), descritos anteriormente.
De cara a nuestra solución podría valer más de una método de perfilado.

En nuestro caso se ha seleccionado el método de K-MEANS por tener un menor coste computacional, por ser el más implantado y las comparativas de rendimiento y efectividad son superiores.
No nos vemos afectados a las limitaciones anteriormente descritas, dado que conocemos a priori el número “k” y verificamos que los clusters que obtenemos tras una primera prueba, no forman una estructura de cluster convexa (véase Figure 9), y podemos eliminar los outliers previo a la tarea de procesado.
En base a estas premisas, podemos utilizar este método de particionado.

Una vez definido el método de particionado, realizamos distintas pruebas para obtener la escala de datos idónea, el valor de “k” (número de clusters), y el número de iteraciones.

De igual forma que el caso de procedimiento propio, se pretende extraer un log y comprobar que se perfila de igual forma.

Por último comprobar que los resultados obtenidos en el perfilado grueso, son similares y con poca tasa de error en comparativa con el uso de la metodología de particionado basado en K-Means.

Para describir el ámbito de requerimientos computacionales (ya que se va a requerir varias iteraciones de comparación entre nodos), se indicará el orden de complejidad de las metodologías utilizadas) para que posteriormente pueda hacerse una similitud con otras investigaciones.

ALGORITMOS SELECCIONADOS

DEFINICION PROCEDIMIENTO PROPIO

Perfilado grueso, es el nombre que hemos asociado a nuestra solución metaheuristica[3] .

Se basa en una simplificación de los procedimientos [3] de perfilado jerárquico, en donde no vamos a tener que mantener los datos acumulativos para gestionar los clusters y una comparativa adicional de los datos acumulativos con respecto a los nuevos puntos.
Pretendemos bajar el tiempo de computación:

Técnica aglomerativa de cluster jerárquico general: $O(n^2)$

CURE $O(n^2 * \log n)$

a una solución del orden: $O(n * \log n)$

Tal como implementan alguno de estos métodos se realizan transformaciones para realizar las comparaciones a través de distintas distancias.
Dado que nuestro fin es perfilar los nodos existentes, y ser capaces de simular el comportamiento de un nuevo nodo, este procedimiento lo consideramos adecuado.

En el proceso de pre-procesado de los datos se ha establecido un vector de bits con la transformación binaria del %average de funcionamiento.
Estableciendo un a 1 si el nodo ha estado activo (con datos acumulativos) más del 75% de la hora y definiendo como 0 en el caso contrario.
Esto genera un vector de bits de 168 posiciones para cada uno de los nodos.
De forma sencilla podemos establecer que nodos están continuamente activos y funcionales (con una tasa de actividad por hora superior al 75%) y cuales no.

[3] Metaheuristica es la combinación de varios métodos heurísticos a alto nivel, para conseguir una búsqueda en el espacio de búsqueda propio, que pueda optimizar los métodos generalistas.
Estos algoritmos son aproximativos (no garantizan la obtención de la solución optima, son relativamente sencillos, y la regla de selección se basa en el instante del proceso y de la historia hasta ese momento)

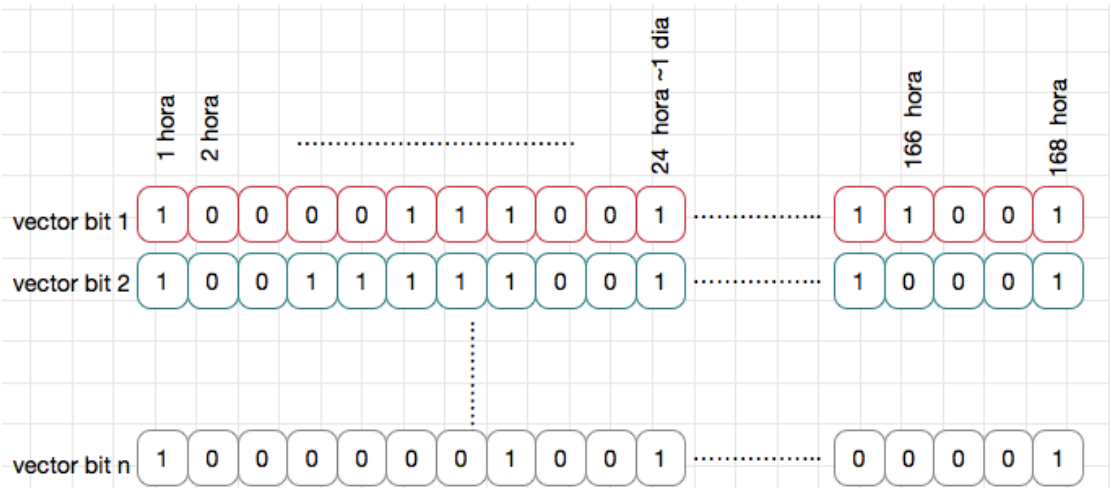


Figure 2 Vectores de bits

$$actividad = \sum_{1}^{168} x_i$$

Nota: Estos rangos de funcionamiento se han dejado parametrizados para poder variarse en valor o eliminarse perfiles.

Nota: Los valores de continuamente activo e inactivo, se tendrán en cuenta para la eliminación de outliers del proceso de K-Means.

Entre cada una de los perfiles, nos podemos encontrar que para un mismo número de bits activos, ejemplo entre 90 a 100 horas de actividad al mes, podemos encontrarnos que esta situación se puede dar en nodos que estén:

3 horas por la mañana activos durante el mes.

3 horas nocturnas durante el mes.

90 horas continuas de actividad…

A través de esta solución, podemos contar las horas de actividad y con ello asociar estos nodos a diferentes perfiles de funcionamiento, esto nos lo determinará el número de horas (x) de cada array de 168 posiciones.

Se ha establecido un rango

Tabla 5 Perfiles asociados al método propio

Valor de estabilidad	Perfil	Rango
x	Estable	$121 \leq x < \leq 168$
	Semi Estable	$81 \leq x \leq 120$

	Poco Estable	21 ≤ x ≤ 80
	Inestable	20 ≤ x

Por tanto para poder hacer una similitud de funcionamiento un perfilado interno, usaremos la distancia de Hamming para ver como un nodo se parece más o menos en base a su actividad, pudiendo agrupar estos nodos que tienen valores pequeños en la distancia de Hamming en un nuevo sub-perfil.

Se han establecido realizar el cálculo de las distancias de Hamming de forma independiente a cada perfil de funcionamiento, ya que mejora sustancialmente los rendimientos de cálculo.
Para el caso de 30 nodos distintos, se mejora de 90 comparaciones en la distancia de Hamming (uno con respecto a los demás) a 21.
El coste computacional de comparar estos valores es del orden:
`O(n * log n).`
Donde estamos comparando sólo los elementos del mismo perfil, siendo el peor de los casos, cuando todos los nodos se encuentran en el mismo perfil.

A demás nos permite de disponer de las horas concretas entre nodos semejantes (objetivo con los cuales queremos comparar), las horas de diferencias a través del offset y size.

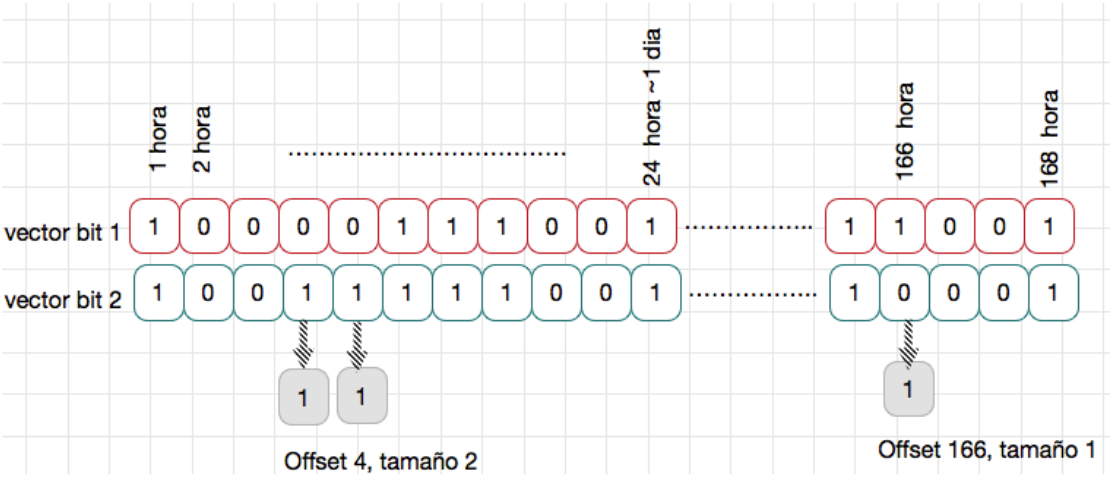


Figure 3 Método de comparación de vectores con la distancia de Hamming

Existe una alternativa que puede mejorar el perfilado, se basa en no hacer la transformación binaria, o simplificación booleana (requerida por Hamming). En este caso truncaríamos el %average a dos decimales que correspondiente al su valor real de uso.
Estos valores son traducidos a cadenas de caracteres (string casting), e implementar una comparación de caracteres. Esto lo implementaríamos con la distancia de *Levenshtein.*
Esta metodología nos haría definir un pre-análisis para determinar los outliers, pero en contrapartida nos mostraría aproximaciones más finas.

EXPLICACIÓN DEL PROCEDIMIENTO POR PASOS

Los datos aquí gestionados se han realizado para 1000 líneas de logs, y posteriormente para 10.000 líneas de logs, no siendo significativos los tiempos de procesamiento.

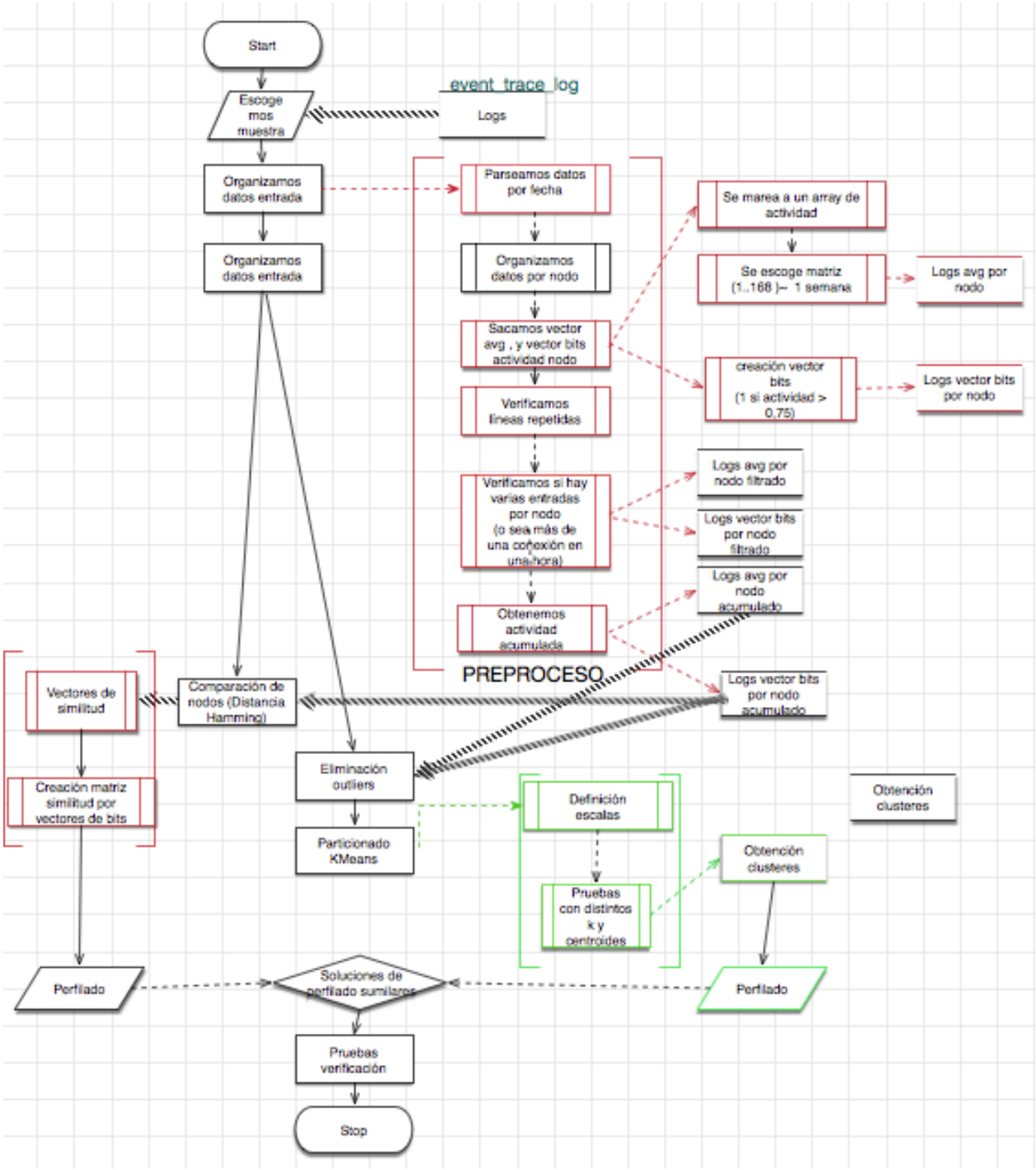


Figure 4 Síntesis del procedimiento de análisis

Nota: Los apartados marcados en rojo, se corresponden a desarrollos realizados en Perl, mientras que los apartados definidos en verde, son desarrollos en Python (ya que Perl carece de la potencialidad estadística ni dispone de librerías que aporten estas funciones).

Extracción de datos:

En un primer momento, se extrae un fichero origen de los logs, hemos de tener en cuenta los campos que son representativos de interés para nuestra gestión.
El nombre del nodo, la fecha de inicio de actividad y fin de la actividad (ambas en formato epoch, o denominada fecha unix que es el número de segundos transcurridos desde 1 de Enero 1970 00:00:00 GMTo ISO 8601 1970-01 T00:00:00Z).

Previo al pre-procesamiento de los datos, hemos seleccionado el tamaño o histórico a analizar para ver el comportamiento de nuestros clusters.

Como veremos posteriormente se ha realizado un procedimiento que hemos denominado DEFINICION , y que ha condicionado nuestra decisión.
De la línea de actividad el análisis se van a analizar las horas de actividad de cada uno de los nodos. Calculando la actividad media de funcionamiento que tiene cada uno de los nodos durante cada hora, siguiendo un procedimiento similar [12].

Nota: Se podría haber escogido una medida distinta como “medias horas” para tener un mejor ajuste de tiempo.

En el análisis queremos ver el comportamiento de los nodos a lo largo del tiempo, por lo que un día nos generaría un array de 24 entradas, 2 días 48, 168 para una semana y 744 entradas para un mes.

Dado que el sistema va a hacer un perfilado de datos masivos y comparaciones sucesivas (dependiendo de método utilizado) se ha escogido la medida de 168 bits de cara a las pruebas.

Este valor se ha parametrizado para poder escoger otra escala.

Pre-procesado de la información:

De esta tarea se encargará un conjunto de scripts en Perl:

`1-array_week.pl:`

Su función es leer el fichero de logs y generar :

- un nuevo fichero con los datos de actividad de cada nodos, y con un formato de (Año, mes, día, hora, minuto) de actividad (parada-comienzo), se extrae una semana completa de actividad. Comenzando el mismo día de la semana extraído por los logs. Generará el fichero salida_vector_avg.txt (destinado al análisis de K-Means).
- Un fichero similar pero con el ajuste binario, creando el fichero de salida vector_array_bist.txt (para el procedimiento de perfilado grueso).

Nota: Esta adecuación se basa en que podríamos modificar el algoritmo para el análisis de un día, o mes, comenzando siempre el análisis un Lunes.

Esto nos alteraría el tamaño de bits a almacenar en los arrays.

Este procedimiento se ha seguido para proteger líneas repetidas en el fichero origen de los logs.

`2-correlate-avg.pl:`

Su función es prevenir líneas duplicadas y eliminar los outliers definidos como aquellos nodos sin actividad o siempre activos.

Adicionalmente:

- Agrega los promedios de los nodos que tengan más de una entrada en la misma hora.
- Ordena y elimina los nuevos outliers tras acumular los nuevos promedios de actividad para los ficheros de %average y vector de bits.

Genera varios ficheros de trace y los ficheros:

- Outpunt_avg_wihout_outliers.txt
- Vector_bits_without_outliers.txt
- Outliers.txt (que nos lista los host que no vamos a procesar).

`3-prepare-for-kmeans.pl:`

Su función es adecuar el fichero con el vector de 168 posiciones con el %average que generará un fichero con el formato requerido para el procedimiento de K-Means.

- Nos muestra los valores de media, varianza y desviación estándar de cada vector de bits y el número de ellos que se aleja de los límites superiores e inferiores.
- Nos aporta, la media, varianza y desviación estándar del conjunto total de datos del análisis.

`5-density.pl`:

Nos calcula la densidad de los arrays.

ÁNALISIS CON PROCEDIMIENTO PROPIO

Una vez preprocesado los datos vamos a calcular los posibles nodos que se encuentran los nodos en base a una tabla de actividades **Error! Reference source not found.** que nos determinan ya los posibles valores de los clusters.

Para ello partimos del fichero de salida preprocesado *Vector_bits_without_outliers.txt.*

Esto se realiza directamente con el script de perl:

`4-hamming-distance.pl:`

Ejemplo (fichero ordenado con la matriz de actividad fichero de entrada):

```
100203236 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
1 0 0 0 0 1 1 1
….
```

```
100205414 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 1
```

…

Nota: Los ejemplos aquí mostrados se basan en el análisis de un fichero de 1000 líneas de logs.

Comparación de vectores de bits:

```
Comparamos 100067629 poco-estable 37: con 100066658 estable 162:
01111111111111111111111101111111111011111111111101111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111110111111111111111111111
11111000000000000000111111111000000000000000001011110000000000000
00000000000000000000000000000000000000000000000011111111100000
0000000000111111100000000000000000111
----------------------
Comparamos 100075173 estable 166: con 100067629 poco-estable 37:
11111000000000000000111111111000000000000000001011110000000000000
00000000000000000000000000000000000000000000000011111111100000
0000000000111111100000000000000000111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111101111111111111111111111111111111111111111111
11111111111111111111111111111111111
----------------------
```

Al realizar la comparación basada en los perfiles, reducimos de para nuestros hosts () de 91 comparaciones:

```
1: D comparamos 0 y 1 100066658 estable:162 con 100067629 poco-
estable:37
2: I comparamos 0 y 13 100066658 estable:162 con 100427449 poco-
estable:38
3: I comparamos 0 y 12 100066658 estable:162 con 100423635 poco-
estable:74
4: I comparamos 0 y 11 100066658 estable:162 con 100422926 semi-
estable:115
```

```
5: I comparamos 0 y 10 100066658 estable:162 con 100413314 semi-
estable:106
….
90: I comparamos 11 y 13 100422926 semi-estable:115 con 100427449
poco-estable:38
91: D comparamos 12 y 13 100423635 poco-estable:74 con 100427449
poco-estable:38
```

A 21 comparaciones y obtener offset más pequeños.

```
100075173 us 100272294 de tipo estables
diff offset 9,size de diff 2
diff offset 12,size de diff 7
diff offset 33,size de diff 1
diff offset 62,size de diff 2
diff offset 77,size de diff 8
diff offset 87,size de diff 1

100075173 us 100208668 de tipo estables
diff offset 47,size de diff 1
diff offset 83,size de diff 1
diff offset 87,size de diff 1
diff offset 107,size de diff 1
diff offset 161,size de diff 1
```

Nota:

Offset 0 a 24, corresponde a las diferencias del primer día.
Offset 25 a 48 , corresponde a las diferencias del segundo día.
Offset de 49 a 72, corresponde a diferencias del tercer día
Offset 73 a 96, corresponde a las diferencias del cuarto día.
Offset 97 a 120, corresponde a las diferencias del quinto día
Offset 121 a 144, corresponde a las diferencias del sexto día.
Offset 145 a 168, corresponde a las diferencias del séptimo día.

De aquí podemos ver los cambios de actividad por día (en el caso que se precisase en un análisis más fino).

El análisis de similitud de un nodo a otro es sencilla, y nos permite conocer el vecino más cercano y la

Matriz de similitudes:

Tabla 6 Matriz de similitudes de nodos estables

Nodos estables				
	100066658	100272294	100075173	100208668
100066658	0			
100272294	25	0		
100075173	6	21	0	
100208668	9	22	5	0

Tabla 7 Matriz de similitudes de nodos poco estables

Nodos poco estables						
	100067629	100427449	100423635	100408768	100389873	100205414
100067629	0					
100427449	56	0				
100423635	49	21	0			
100408768	66	56	63	0		
100389873	53	51	42	53	0	
100205414	42	64	75	60	71	0

Tabla 8 Matriz de similitudes de nodos semi-estables

Nodo semi estables				
	100203236	100422926	100413314	100309685
100203236	0			
100422926	76	0		
100413314	81	71	0	
100309685	79	73	100	0

Apreciamos que los valores de similitud en los nodos poco estables o semi estables son muy elevados, por lo que hay poca cohesión en el grupo. No así en los nodos estables.

Con otro tipo de agrupación (por días) o variando los umbrales de las tipologías podríamos adecuar estos valores.

A modo de ejemplo se muestra para el entorno semi-estable en las primeras 72 horas (esto saldría de forma directa variando el tamaño del array a gestionar de 168 a 72):

Tabla 9 Matriz de similitudes de nodos semi-estables en las primeras 72 horas

Nodo semi estables				
	100203236	100422926	100413314	100309685
100203236	0			
100422926	37	0		
100413314	39	23	0	
100309685	41	28	37	0

Nota: Opcionalmente: se puede realizar la "Matriz de similitudes por día".

Vemos que aunque la actividad global de cada uno de los nodos se mantienen dentro del rango descrito (81 a 120 horas al mes), durante los primeros 3 días, las horas de actividad varían un 50% de las veces.

La gestión de (crear, almacenar, actualizar) las matrices de similitud para calcular el vecino más próximo conlleva unos costes computacionales altos cuando el número de elementos es alto (de ahí nuestra segmentación previa en perfiles), pero cuando esto ocurre se suelen emplear otros métodos de agrupamiento basado en los algoritmos de particionamiento.

Calculamos nuestros tiempos de computación:

Tabla 10 Comparaciones y tiempos de computación

Líneas de logs procesadas	Comparaciones requeridas	Tiempo de computación
1.000	21	0,08 segundos
10.000	91 / 34 optimizado	0,6 segundos
100.000	739	11 segundos

En nuestra metodología, hemos realizado un ajuste previo (transformando nuestro vector a un array binario) y un pre-ajuste de tipo en base a la actividad mensual, posteriormente mediante la distancia de Hamming realizamos la comparación de arrays binarios para hacer una comparativa entre nodos y un perfilado más a medida (que nos permitiría disponer de las matrices de similitud), que de forma más sencilla y con coste de computación más bajo nos ayudaría a identificar el perfilado que buscamos.

Vamos a comparar esta metodología r con los resultados que nos propone el método K-Means para verificar que se cumple y que podemos utilizar el perfilado a medida.

ANÁLISIS CON MÉTODO DE PERFILADO K-MEANS

Estamos en un sistema indeterminista (desconocemos a priori, si el usuario va a arrancar el ordenador, si aún así va a levantar el nodo), si no lo ha hecho durante los días anteriores por distintas causas personales, o si ya no va a usar el entorno. Por lo cual, la predicción o teorías como Montercarlo quedan fuera de este análisis.

Podríamos analizar la covarianza (relación entre variables) con otros datos posibles extraídos de los logs como uso de la CPU, almacenamiento, etc, pero no está en e la definición de la investigación.

En nuestro caso tampoco se pretende hacer una predicción del comportamiento, para un número elevado de nodos, sino únicamente ver el comportamiento que se han seguido con lo logs y ver la lógica que se ha seguido, Por tanto no haremos una regresión logística de los PCA (Principales Componentes de Análisis). Esta es la metodología básica que se suele emplear en Machine Learning para hacer predicciones.

Nota: PCA se utiliza para descomponer un conjunto de datos multivariados en un conjunto de componentes ortogonales sucesivos que explican una cantidad máxima de la varianza. Si utilizamos scikit-learn, PCA se implementa como un objeto transformador que aprende de “n” componentes en su método de ajuste, y puede usarse en nuevos datos para proyectarlo en estos componentes.

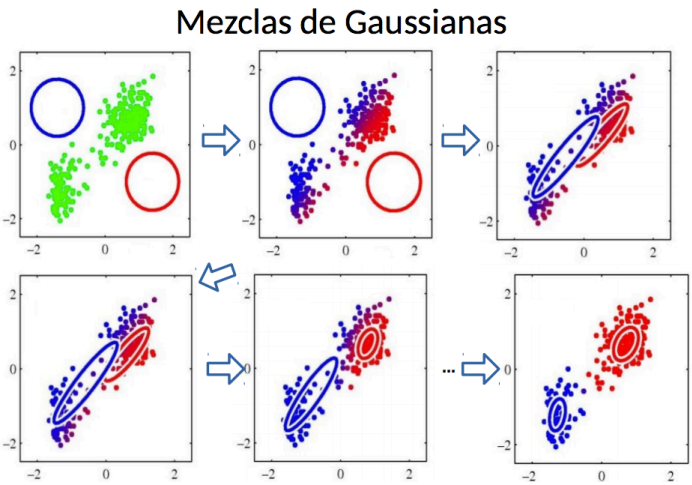


Figure 5 Asimilación que el cluster sigue una distribución normal

Para este análisis como vamos a utilizar el método de particionamiento de KMeans [11], hemos filtrado y eliminados los valores acumulativos en una hora superiores (valore de y > 0,98) por considerarlo un nodo muy estable (siempre activo), pudiéndose haber conectado y desconectado varias veces en la misma hora, pero de forma global activo.

De igual forma se han eliminado aquellos nodos cuya actividad acumulada en una hora no ha sido superior a 0.01%, por suponerse como nodo inactivo (barras oscuras).

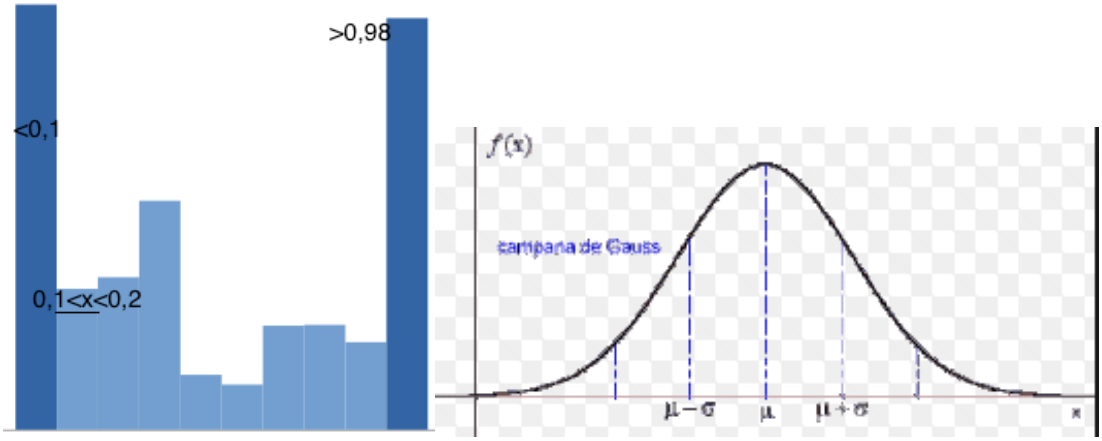


Figure 6 Densidad de %avg para 10.000 líneas de logs

Con un valor de media: 0,47

Varianza: 0,14

Desviación estándar: 0,38

No se ha realizado un ajuste o simplificación de datos a un entorno binario (activo >0,75% del uptime) e inactivo en caso contrario, como se hizo en el planteamiento de perfilado grueso, ya que obtendríamos como resultado dos líneas horizontales, generándonos dos clusters muy marcados.

Aun así, indicar que esta transformación suele ser utilizada en sistemas en los que es difícil establecer puntos bidimensionales, por lo que se relaciona la función a un “cumple” o “no cumple”, mapeando a un valor booleano. Y a partir de esta transformación, calcular la distancia euclídea para el calculo entre puntos bidimensionales por ejemplo entre la distancia de un punto o nodo y la distancia del centroide del cluster para medir la distancia intra-node.

El uso de la distancia euclídea en valores binarios (existe o no existe) o en nuestro caso está activo o no está activo en un momento determinado, pudiendo usar la fórmula $D_{ij} = \sqrt{(b+c)}$

Tabla 11 Adecuación de valores de K-Means (valores no numéricos)

Casos	j	
	1	0

i	1	a	b
	0	c	d

“a”: total de atributos presentes en el caso “i” y caso “j”.
“b”: total atributos presentes en caso “i” y que no están en el caso “j”.
“c” total de atributos que no están en “i” y que están en “j”.
“d:” total de atributos que no se encuentran en ninguno.

De esta forma podemos usar el coeficiente de “concordancia simple”.

En nuestro caso, no es necesaria esta transformación (ya utilizada anteriormente), porque disponemos del vector de %average, que nos propone valores más ajustados y son válidos para el cálculo de la distancia entre nodos.

Para el agrupamiento en K-Means se ha desarrollado un script en Python: `6-create-kmeans.py`:

Genera los clusters, a través de un fichero de entrada con los valores de los puntos siendo:

X: hora de la medida (array de 168 posiciones).

Y: %average de actividad del nodo.

Genera los ficheros de logs de los datos (de cara a las comprobaciones).

Dibuja los gráficos de los puntos y los clusters.

Nota: Se ha utilizado Python por su capacidad de utilizar librerías estadísticas y de perfilados de ejemplo.

Distancia intra e inter clusters:

Cuanto mayor sea la distancia inter-clústeres, más sencillos será el perfilado, ya que esto mostrará que los clusters están realmente separados unos de otros. Por otro lado, la minimización de la distancia intra-clúster es mejor porque muestra que los conglomerados están más unidos o cohesionados.

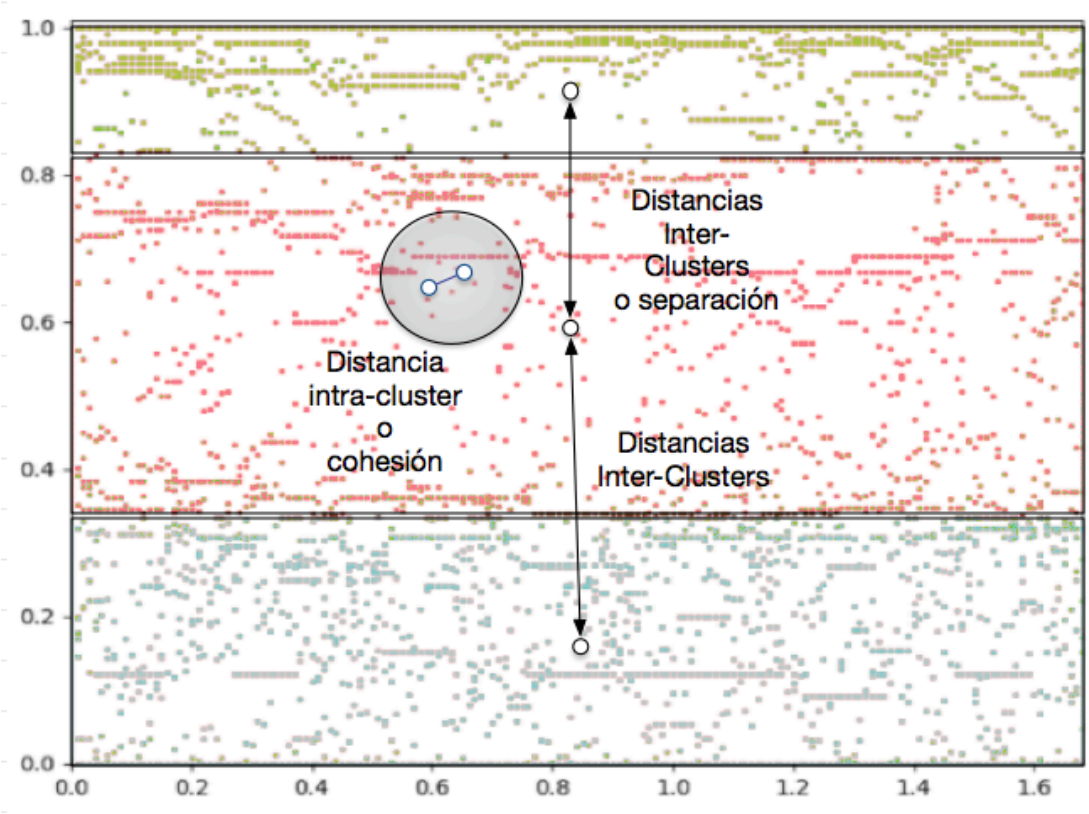


Figure 7 Definición de distancias intra e inter clusters

Véase fórmulas 2 y 3.

Por tanto, realizaremos un ajuste previo de los datos, que agruparemos en varios datasets, para luego implementar directamente el método de perfilado de K-Means.

Para realizar esta implementación, requiere que tengamos dos coordenadas en nuestros datos.

Actualmente tenemos un conjunto de nodos. 10005313, 10000331,…, con sus actividades que hemos descompuesto en horas.

En la selección de los valores (bidimensionales) para tener puntos (x,y) donde “x” será la hora del mes (0..168) e “ y” el %actividad del nodo“, para poder representar el histograma de actividad de todos los nodos, y analizar si dentro de la solución de valores por hora de actividad general se crean cluster donde podamos analizar si un nodo pertenece a ese cluster en conjunto horario (mañana, tarde noche), da valores incorrectos.

En los resultados obtenidos se aprecia, que es independiente del número de clusters que se seleccionen o el número de iteraciones que se realicen para el ajuste de la distancia de un nodo al centroide más cercano (distancia intra-cluster).

Esto es debido a que cuando vamos aumentando las “y” que crecen de 0 a 168 , se distorsiona el valor de la distancia euclídea
Ejemplo Este ejemplo se ha probado con k=2, k=3, k=4, número de iteraciones Iteraciones,=2, Iteraciones=20, Iteraciones=200, 500 y 2.000
En el desarrollo, k o el número de clusters, se coge del número de centroides definido.
Es independiente del posicionamiento de los centroides, ya que lo que nos determina la distorsión es el crecimiento iterativo de la “x”.
En nuestra explicaciones con el fin de hacerlo entender, seleccionamos los centroides al azar:

Centroide0 =[64,0,32]
Centroide1 =[75,0,96]

Como el array comienza de 1 a 168 en los primeros ajustes de los puntos [1,0.942] , la distancia euclídea será 63,003 con el Centroide0, y 74. 00 con el Centroide1, colocando este punto dentro de un cluster.
En este caso seleccionamos la distancia mínima (cohesión) o cercanía al centroide, asociando temporalmente (hasta la siguiente iteración) el punto en este cluster.
Con el fin de limitar fallos, el nuevo centroide será, la media entre los dos puntos [32,5 0,6305]
Para el siguiente punto del mismo nodo, (siguiente hora), el comportamiento del nodo es similar, y por tanto similar su actividad;

[2, 0,927]

Ahora deberíamos ver su distancia comparándolo con:

Centroide0 =[32,0,63]
Centroide1 =[75,0,96]

Siendo la distancia mínima otra vez con el Centroide0.

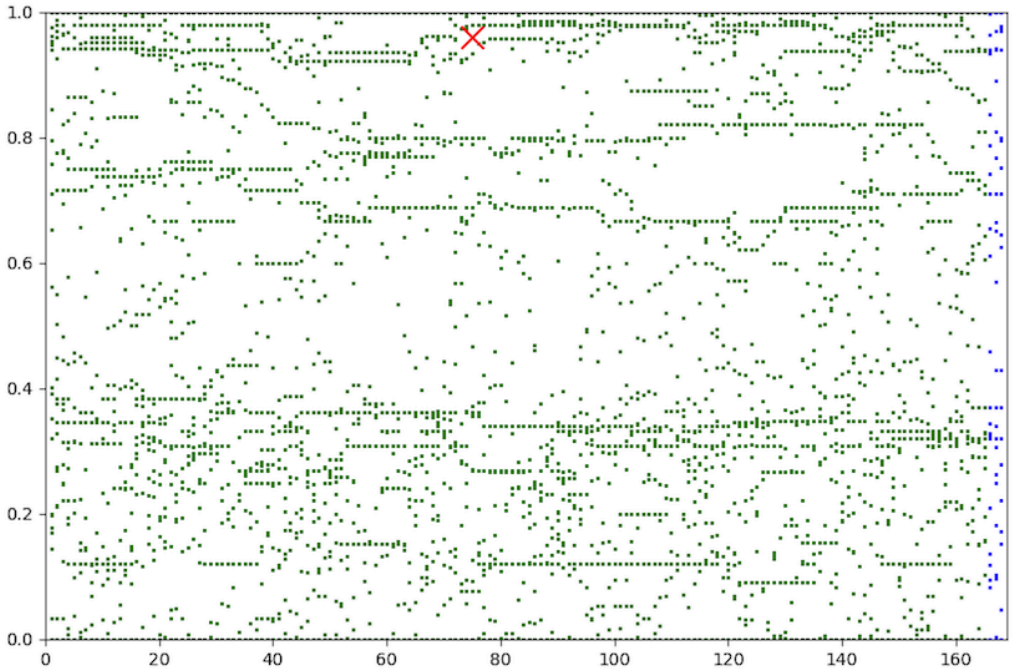
Solamente para valores de puntos altos de x (horas cercanas a final de mes) con valores altos de activad, se genera un cluster en la parte de la derecha, que puede ser casi despreciable

Resultados:

Para K=2 (independiente del número de iteraciones):

Cluster0: 62.370 puntos (verde)

Cluster1: 1.134 puntos (azul)



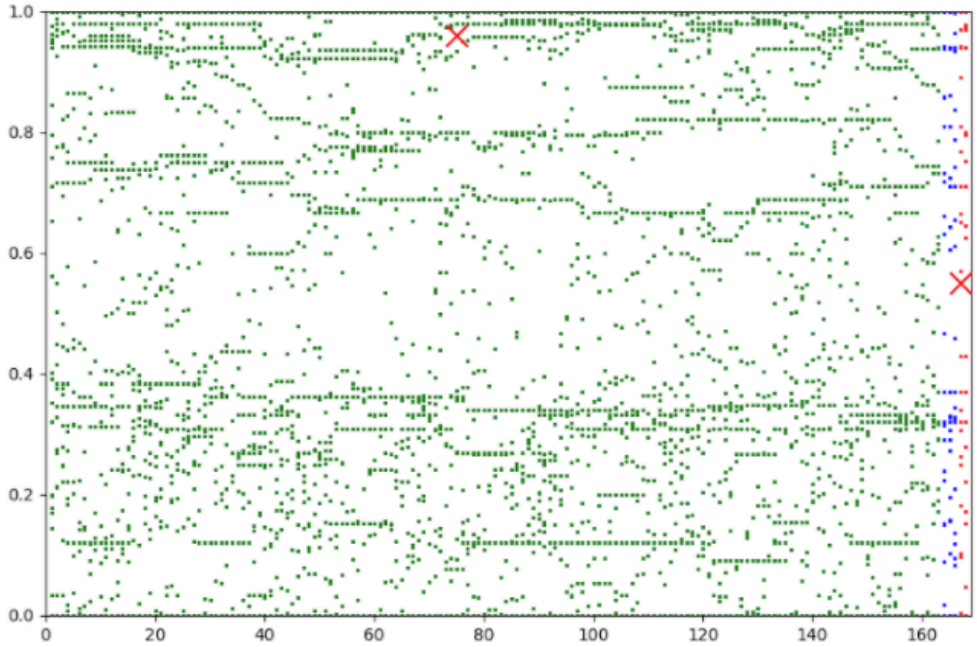
Para K=4 (independiente del número de iteraciones)

- Resultados:

Cluster0: 61.614 puntos (verde)

Cluster1: 1134 puntos (azul)

Cluster2: 756 puntos (rojo)



Según vamos avanzando en el array, siempre estará cerca de este valor,

$$D_{ij} = \sqrt{(X_2 - X_1)^2 + (y_2 - y_1)^2}$$

X_2 crece más rápido y la distancia será siempre peor.

Veamos que es un problema de escala, y dividimos el valor de la x.
Redefinimos la escala, dividiendo la escala de x, entre 100, para que no nos afecte, y obtenemos los siguientes resultados.
Pasando a ser el rango de x (0 a 168) a un rango (0, 0.01 ... 1,68), manteniendo una escala similar a “y” (0 ..1).

Repetimos el procedimiento y con distinta escala para k a 2, y obtenemos los siguientes resultado:

Encontrándonos dos clusters:

Seleccionamos el valor de los centroides coincidentes con nodos:

- Valores de partida:

Centroide inicial 0: [0,03 0,94] (correspondiente al nodo [1 17])

Centroide inicial 1: [0,78 0,92] (correspondiente al nodo [11500 71])

- Resultados obtenidos:

Cluster 0: 31.752 puntos (con y< 0,92)

Cluster 1: 31.752 puntos (con y>0,92)

Centroide final 0: [1,67 0,094]

Centroide final 1: [1,669 0.647]

Apreciamos que en los resultados, el número de elementos en los clusters es similar tamaño, tras variar la escala.

Este mismo experimento, con k=2 y aumentando el número de iteraciones a 20 para el ajuste de la cohesión (variable de código repeticiones = 200) y sin variar los centroides, definidos obtenemos exactamente los mismos valores. Luego el ajuste de distancia de centroides no ha variado significativamente.

Nota: No se muestran los gráficos por ser los mismos.

Realizando esto mismo, con otros centroides escogidos por valores al azar:

- Valores de partida:

Centroide inicial 0: [0.75 0.96]

Centroide inicial 1: [0.45 0.168]

Repeticiones: 20

- Resultados obtenidos:

Cluster 0: 31.572 (con x(0 ..1,68 **, y< 0.922**)

Cluster 1: 31.572 (con x (0 .. 1,68 con **y> 0.922**)

Centroide final 0: [1,67 0,094]

Centroide final 1: [1,669 0.647]

Cluster 0:

Media[4]: 0,667

Varianza:[5] 0,228

Desviación estándar[6]: 0,47756

Cluster 1:

Media: 0,875

Varianza: 0.237

Desviación estándar: 0,4866

[4] Media aritmética: $\bar{x} = \frac{1}{n} * \sum_{i=1}^{n} x_i$

[5] Varianza: a qué distancia los datos están agrupados alrededor de la media.

$$\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

[6] Desviación estándar: cuán dispersos están los números en la muestra de datos. $s = \sqrt{\sigma_n^2} = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2}$

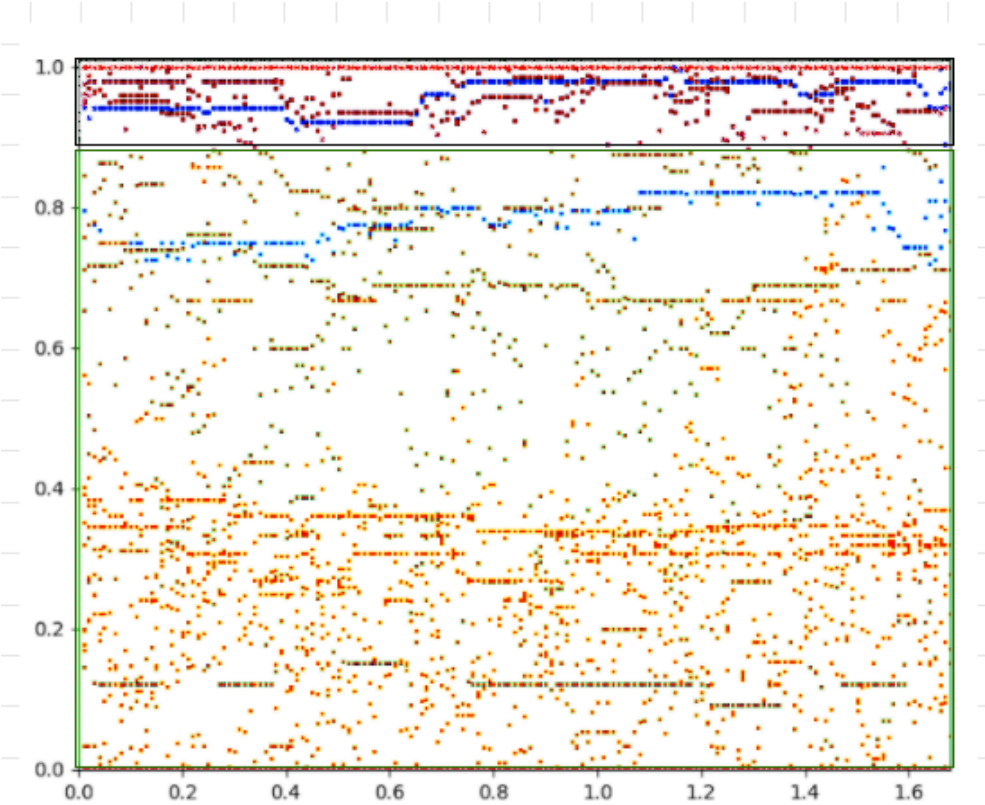


Figure 8 KMeans perfilado de logs para k=2

Para k=2, , centroides iniciales aleatorios, y número nodos 4.676 nos arroja los siguientes resultados:

Cluster 0: 2588 ,media: 0.467, varianza=0,231, desviación típica: 0.480
Cluster 1: 2088 ,, media= 0,96, varianza 0,212 , desviación típica:0,460
Que hace que los puntos estén más ordenados.

Para 63.144 nodos, aparecen puntos intercambiados porque el valor de la desviación típica es muy elevado y superior a la distancia de las medias.

Sin embargo de forma general, para k=2, el sistema se comporta como el análisis de perfilado con comparación basada en la distancia de Hamming, pudiéndose haber usando la equivalencia de 0, 1 a activo o inactivo. En este caso, si el porcentaje de actividad de un nodo en una hora determinada está

cercana al valor inferior marcado por el cluster 0.97 estará en ese cluster y si su actividad media en esa hora es superior a 0.922 estará en el cluster 1.
Lo normal es que un nodo tenga una comportamiento similar.
En caso de no ser así, pasaría a formar parte del cluster0 o cluster 1 en función de su tanto % de uptime.

Cluster 0: 22.377 puntos
Cluster 1: 22.421 puntos
Cluster 2: 18.706 puntos

Con esto vemos que se distribuyen en nodos de forma

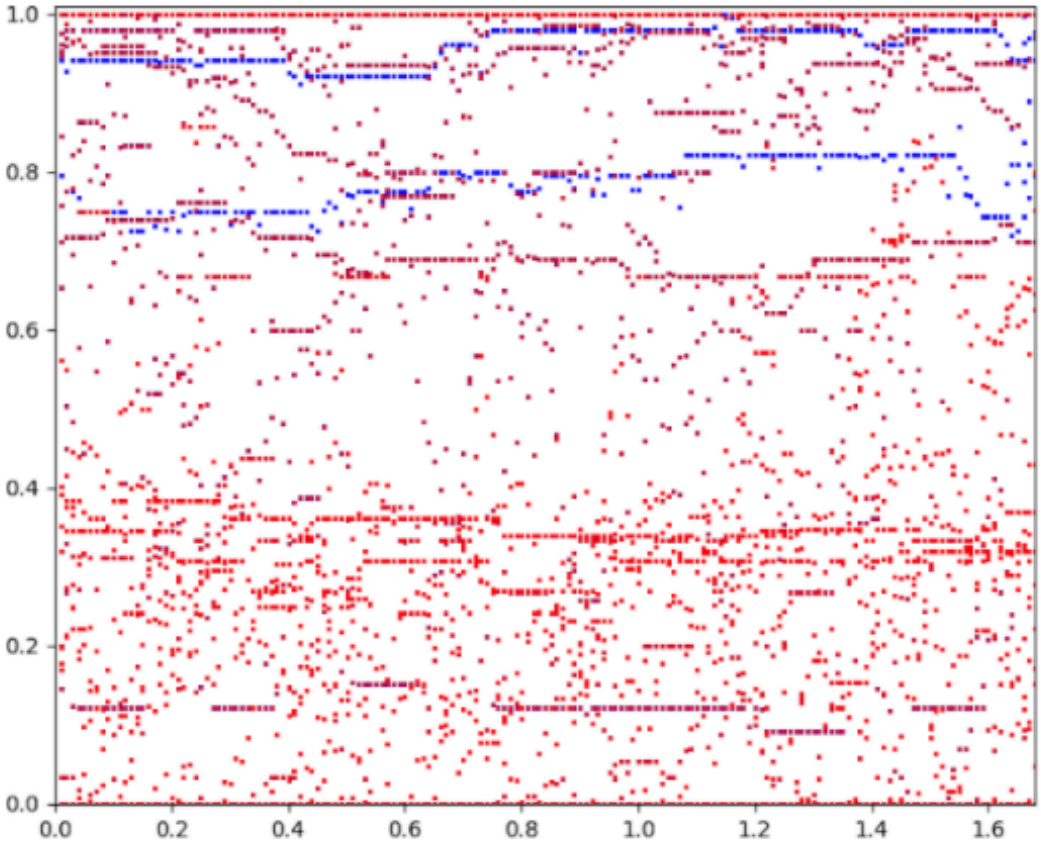


Figure 9 KMeans clusters de logs para k=3

Nota: En la parte superior hay puntos repetidos y no se aprecia la densidad.

Aquí veos que los clusters se han centrado, en líneas horizontales.

Estos resultados , los comprobamos poniendo distintos centroides, y variando el valor de k, ya que K-Means es sensible a la elección de los centroides (a no ser que tal como se ha indicado más arriba, se realicen los ajustes, y se compruebe con distintos valores de centroides que no existen modificaciones significativas.

Se varia el valor de k=3 y valor de las repeticiones (2, 10, 20, 40).

- Valores de partida:

  Centroide inicial 0: [0.75 0.96]

  Centroide inicial 1: [0.45 0.168]

  Centroide inicial 2: [0.167 0.55]

  Repeticiones: 20

- Resultados obtenidos:

Cluster 0: 22.421 puntos (valore 0 <y>0,3)

Cluster 1: 18.706 puntos (con y>0,94)

Cluster 2: 22.377 puntos (la mayoría de los punto 0.34~y<=0,97)

Centroide final 0: [1,67 0,094]

Centroide final 1: [1.489 0.99]

Centroide final 2: [1,669 0.647]

El cluster 3 tiene menores intervalos de disponibilidad en la media (solo el 10% de los hosts tienen intervalos de disponibilidad inferiores a 10 horas en promedio), pero la desviación estándar[7] es mucho menor que la que muestra el cluster 2.

[7] Desviación estándar: medida de dispersión, que indica qué tan dispersos están los datos con respecto a la media.

El cluster 1, tiene los mayores valores de disponibilidad y con una desviación estándar más baja.

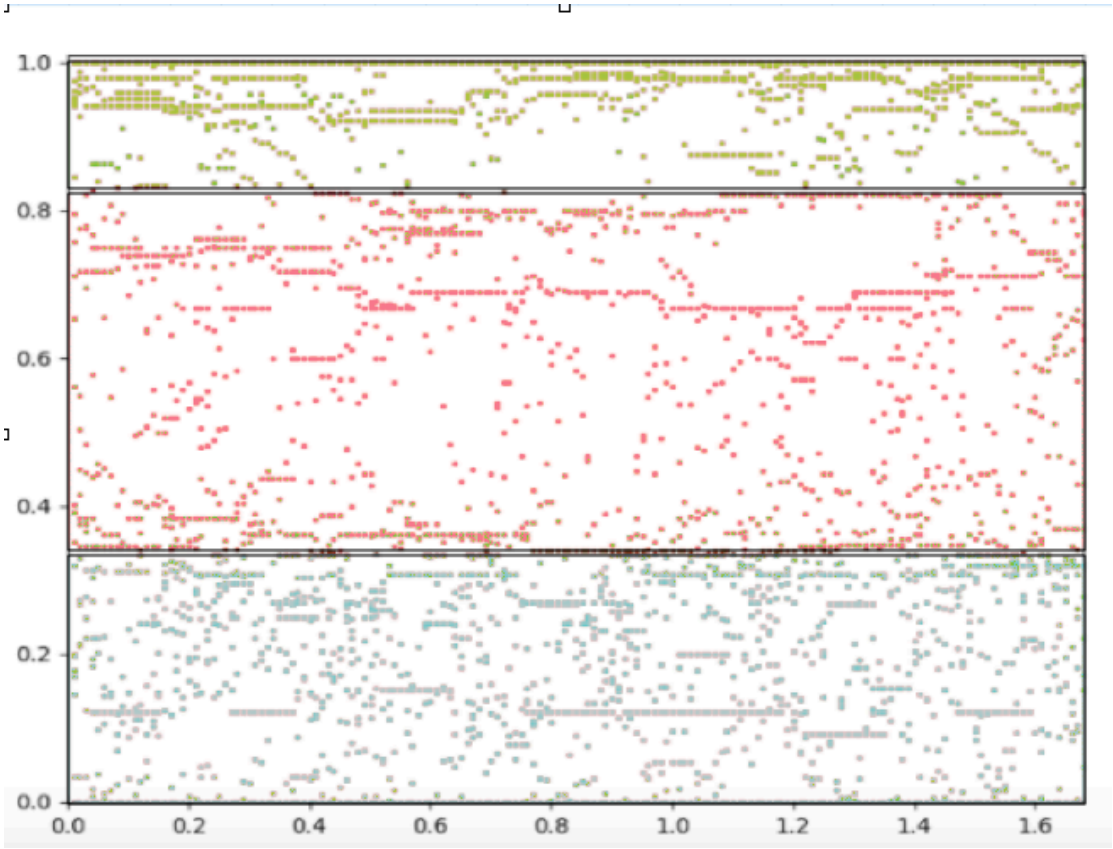


Figure 10 K-Means clústeres logs k=3, agrupado

k=4 y valor de las repeticiones:

- Valores de partida:

  Centroide inicial 0: [0.75 0.96]

  Centroide inicial 1: [0.45 0.168]

  Centroide inicial 2: [0.127 0.55]

  Centroide inicial 3: [0.167 0.88]

  Repeticiones: 20

- Resultados obtenidos:

Cluster 0: 14.876 puntos (1,21<x<1,68, 0,4<y<0,6) (0.01<x<0,168, 0,0<y<9,16)

Cluster 1: 14.604 puntos (0.6<x, y<0,95)

Cluster 2: 15.854 puntos (0<x, y~0,34) (0,4 >x, y~0,97)

Cluster 3: 18.170 puntos (y > 0.98)

Centroide final 0: [1,48 0,99]

Centroide final 1: [1.66 0.64]

Centroide final 2: [1.67 0.96]

Centroide final 3: [1.167 0.097]

Nos muestra una tendencia similar, en bandas horizontales , por lo que se planifica para k=5, se revisa su funcionamiento y se verifica tendencia.
Las pruebas se han repetido, con repeticiones=5, 10 y 20.

Efectivamente, se apreciaba que la escala seleccionada inicialmente no era adecuada.

Una alternativa podría haber sido, modificar la escala y usar la moda[8] (K-Moids) en vez de la distancia euclídea que utiliza K-Means.

Extraemos el resumen:

Tabla 11 Resumen de los datos anteriores

Valores de partida		k	C0	C1	C3	C4	Centroides finales
Numero de puntos: 63.144		2	elementos	elementos	elementos	elementos	[1,67 0.094] [1.669 0.647]
Centroides iniciales	[0.75 0.96] 0.45 0.168]		31.572	31.572	-	-	
Estructura	Lineal		**y < 0.922**	**y > 0.922**	-	-	
Otros datos			Media[9]: 0,667	Media: 0.875			

[8] Moda estadística: se basa en usar el elemento de valor medio tras una ordenación de los componentes.

[9] Media aritmética: $\bar{x} = \frac{1}{n} * \sum_{i=1}^{n} x_i$

			Varianza:[10] 0,228	Varianza: 0.237			
			Desviación estándar[11]: 0,47756	Desviación estándar: 0,4866			
Valores de partida			elementos	elementos	elementos		
Número de puntos: 63.144		3	22.377	22.421	18.706		[1,67 0,094] [1.489 0.99] [1,669 0.647]
Centroide iniciales	[0.75 0.96] [0.45 0.168] [0.167 0.55] [0.167 0.55]						
Estructura	Lineal		**y>0.3**	**y> 0.94**	**0.3<y<0.94**	**-**	
			Elementos	Elementos	Elementos	Elementos	
Número de puntos: 63.144		4	**14.876**	**14.604**	**15.854**	**18.170**	[1,48 0,99] [1.66 0.64] [1.67 0.96] [1.167 0.097]
Centroides iniciales	[0.75 0.96] [0.45 0.168] [0.127 0.55] [0.167 0.88]						
Estructura	Lineal		(1,21<x<1,68, 0,4<y<0,6) (0.01<x<0,168, 0,0<y<9,16)	0.6<x, y<0,95	(0<x, y~0,34) (0,4 >x, y~0,97)	(y > 0.98)	

Valores de nuestros vectores:

Aportamos una serie de valores adicionales que creemos de interés de cara a identificar falsos positivos y definir nuestros datos.

[10] Varianza: a qué distancia los datos están agrupados alrededor de la media.

$$\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

[11] Desviación estándar: cuán dispersos están los números en la muestra de datos. $s = \sqrt{\sigma_n^2} = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2}$

Tabla 12 Valores de los vectores filtrados

nodo	Media	Varianza	Desviación estándar	Nº sup	Nº inf
100067629	0	0.05	0.23	35	0
100117943	0	0	0	0	0
100327214	0	0	0	0	0
100422568	0	0	0.01	19	0
100427449	0	0.02	0.14	40	0
100052112	1	0	0.02	0	30
100265331	1	0	0.02	0	12
100317715	1	0	0.04	0	54
100422926	1	0.21	0.46	0	59
100408768	0.03	0.02	0.12	35	0
100205414	0.09	0.01	0.11	36	0
100389873	0.12	0.04	0.2	44	0
100423635	0.17	0.06	0.24	48	0
100203236	0.22	0.01	0.08	39	18
100413314	0.22	0.01	0.1	40	22
100309685	0.24	0.01	0.1	24	31
100075173	0.31	0	0.03	17	39
100055671	0.35	0	0.02	30	12
100208668	0.35	0	0.07	31	38
100272294	0.48	0.07	0.27	26	21
100252332	0.62	0.03	0.18	26	25
100019206	0.64	0.04	0.19	19	52
100355392	0.69	0	0.03	30	9
100066658	0.78	0.05	0.21	3	45
100224447	0.78	0	0.03	44	20
100233682	0.93	0.01	0.07	0	39
100021662	0.96	0	0.02	3	30
100284574	0.98	0	0.03	0	48

Nº sup: límite superior definido por la media + desviación estándar.

Nº inf: límite inferior definido por la media – desviación estándar.

Nota: Analizamos los campos que superan bien por el por la parte superior o la inferior los valores límite.
Hacemos especial hincapié al nodo 100422926 por disponer una desviación estándar elevada.

De lo que se extrae que:
10006658 son aporta valores con tasas de actividad muy bajas en horas matutinas y muy altas en nocturnas, que nos dan un alto valor de desviación típica, dando una gráfica similar a los puntos azules de Figure 12.
El resto son nodos con muy poca actividad o con mucha actividad que puntalmente durante unas horas han cambiado de modo de funcionamiento.

Vemos que cada uno de los valores de los nodos están muy cercanos a su media a través de la columna de varianza, esto nos indica que los nodos se comportan de una forma muy estable.

POTENCIAL FASO POSITIVO 1

Estos clusters se generan siguiendo un comportamiento previsto, ya que en los logs, una máquina se comporta encendida o apagada un porcentaje de tiempo similar durante una hora. Esto es debido al patrón de asignación de recursos que hace Seti@home (base de nuestro análisis de logs).
Véase Figure 9 KMeans clusters de logs para k=3 donde el cluster azul (correspondiente al nodo2) asignan puntos que se encuentran adicionalmente dentro del rango del cluster 0.
De igual forma ocurre en el resto de las figuras representadas en los demás ensayos.

Si el comportamiento general de los nodos que aportan valores a Seti@home hubiese seguido un comportamiento distinto (uptimes elevados siempre a

horas concretas) los clusters mostrados serían verticales, ya que KMeans depende de los parámetros aleatorios de entrada.

La tendencia funcional de los nodos junto con el gran volumen de información obtenida, crean estructuras de particionamiento horizontales en nuestro caso.

En el caso que se analizase un nodo cuya porcentaje de actividad variase a lo largo de un calendario sólo se activase a horas concretas, mostraría una actividad donde pertenecerá a un cluster u otro dependiendo de su grado de actividad.

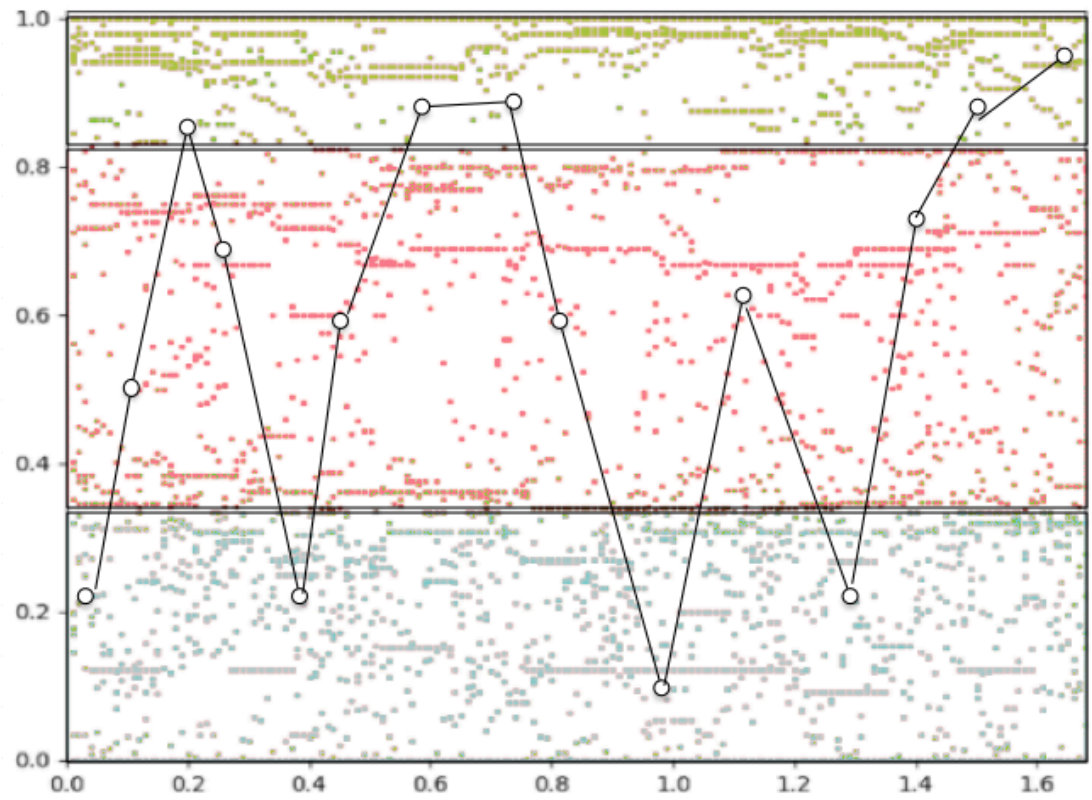


Figure 11 Potencial falso positivo con un nodo con comportamiento muy aleatorio

Si la tendencia de funcionamiento de los nodos fueran similar al comportamiento de este último nodo los clusters generados tendrían geometrías diferentes.

Sin embargo se ha cotejado para 1.000, 10.000 y 100.000 líneas de logs que no se produce en base a su media y desviación estándar.

POTENCIAL FASO POSITIVO 2

Sin embargo aunque los datos aporten los resultados previstos. Podrían darnos otro falso positivo dependiendo del valor “k” o número de clusters. A mayor valor de “k”, mayor es la posibilidad de error. Esto es debido a que estos métodos de perfilado analizan cada uno de los puntos discretos, o de forma individual etiquetándolos dentro de cada cluster o conglomerado, sin tener en cuenta históricos de otros puntos.

En nuestro caso queremos perfilar todos los puntos del array de bits.

Este problema, nos lo encontraríamos igualmente en los clusters basados en métodos de densidad o jerárquicos.

Ejemplo: *podríamos tener “n” arrays comportándose como nodos definidos por los puntos marcados en “azul” y “n” nodos como los puntos marcados en rojo.*

El valor de los clusters para “k=3” podrían ser los mismos, pero los nodos a veces se comportan como el cluster 1 y a veces como el cluster 2.

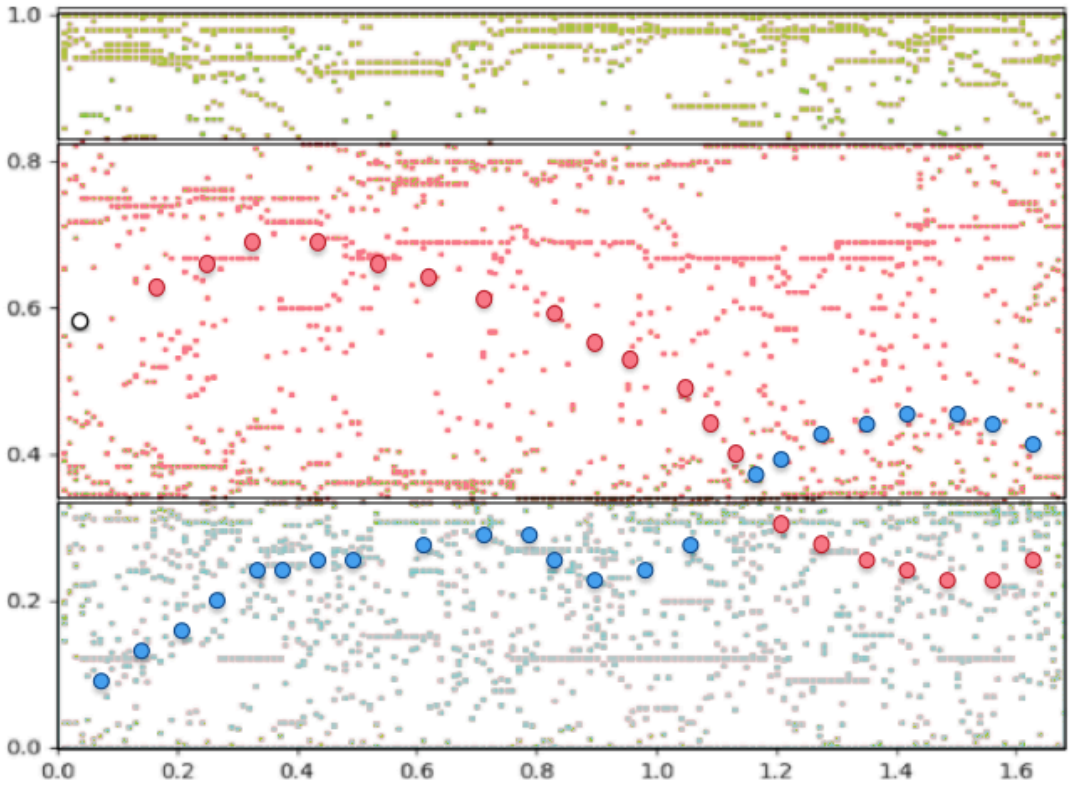


Figure 12 Posible falso positivo de nodos con tendencias contrarias

Realizamos un análisis de los clusters, para comprobar esta situación.

Esto lo vemos en la siguiente tabla:

Tabla 13 Análisis de datos en la asignación de clúster

nodo	media	comportamiento	C 0	C 1	C 2	Mayor	Menor	Observaciones
100019206	0	aleatorio	0	168	0	0.9	0,29	Cambios entre 0,3 y 0,8
100021662	0,96	estables valores-altos	168			0.98	0,22	Casi todos los punto 0,94
100052112	1	estable valores altos	168			1	0,952	Casi todos los puntos 1
100055671	0.35	estable valores medio-bajo		168		0,362	0,319	Casi todos los puntos 0,340
100066658	0.78	aleatorio		151	17	1	0,147	Cambios entre 0,3 y 0.,91 se genera un nuevo cluster 2 para valores bajos
100067629	0	aleatorio		162	6	0,857	0	Cluster 1 (valores 0,5 a 0,7)
100075173	0.31	estable valores medio-bajo		168			0,272	Casi todos los punto 0,34
100117943	0	estable valores muy bajos		168		0	0	Casi todos los puntos 0
100203236	0.22	estable valores medio-bajo			168	0,317	0,195	Valores muy aleatorios entre 0,19 y 0,27
100205414	0.09	aleatorio			168	0,453	0	inestable con muchos valores a 0, y otros entre 0,19 y 0,27
100208668	0.35	estable valores medio-bajo			168	0,444	0,136	Casi todos los valores entre 0,357 y 0,46
100224447	0.78	estables valores-altos	168			0,8	0,725	Casi todos los valores en rango 0,7
100233682	0.93	estables valores-altos	168			1	0,77	Casi todos los valores están en 0,9 y 0,8
100252332	0.62	aleatorio	12		156	1	0,34	valores que varían entre 0,48 y 0,8
100265331	1	estables valores-altos			168	1	0,94	Valores muy estables
100272294	0.48	aleatorio	6	11	151	1	0,17	inestable
100284574	0.98	estables valores-altos			168	1	0,94	
100309685	0.24	estable valores muy bajos		168		0,22	0,17	

100317715	1	estables valores-altos		168	1	0,95	
100327214	0	estable valores muy bajos	168		0.9	0	
100355392	0.69	estable valores medio alto		168	0,75	0,67	
100389873	0.12	aleatorios valores medio bajo	160	8	0,749	0,151	Muchos valores 0,151 y 0,121, asigna cluster 2 para valores 0,7
100408768	0.03	estable valores muy bajos	168		0,407	0	Valores muy bajos
100413314	0.22	estable valores medio-bajo	168		0,314	0,04	
100422568	0	estable valores medio-bajo	168		0,2	0	
100422926	1	estables errático	42	126	1	0,45	Muchos puntos a 1 y otros a 0
100423635	0.17	aleatorio	158	10	0,649	0	cluster 1: muchos valores a 0, cluster 2:valores 0,4-0,6
100427449	0		168		0,5	0	

Nota: Se ha hecho pruebas con iteraciones de reajuste con valores 500 y 2000 de cara a dar la posibilidad de reajustes de la los centroides a nodos.

C0: Corresponde al cluster0, C1, al cluster1, …

Las celdas oscurecidas muestran hosts con comportamientos erráticos que hacen que estén en más de un cluster, como de esperar ya que su desviación estándar era elevada.

Se podría realizar una asignación a partir de identificar el porcentaje de los puntos del array de bits que pertenecen al cluster1, y cuales al 2, y asociar al array de bits a aquel rango donde mantenga su mayor % de actividad, ya que vemos que estos suelen ser puntuales.

Para analizar esta suposición analizamos los puntos de cada vector.

El peor de los casos es cuando el número de puntos sean muy parecidos en varios clusters y lleguemos a una indeterminación de asignación.

MODIFICACIÓN DEL PLANTEAMIENTO DE K-MEANS

En este caso, partimos del fichero de %avg de 168 líneas por nodo.
Queremos establecer el nodo con sus x=[168] posiciones, como elemento (ente a analizar), siendo cada una de las posiciones una característica (feature) del nodo.

La selección de características [10], puede resultar de mayor repercusión en el error de clasificación:
Pocas características, nos dan regiones de decisión más simples.

Las características deben constituir agrupaciones de valores medios que se mantienen muy semejantes entre objetos de la misma categoría y diferentes entre objetos de distintas categorías.
En nuestro caso son %average de cada hora, que coincide con esta definición.

Para este ensayo se va a generar un nuevo script de Python k-meanas-feature_vector.py haciendo uso de la librería de skearn (skearn.cluster) para importar el módulo ya creado de K-Means. También usaremos sklearn.decompostion para importar el módulo de PCA.

Las pruebas aquí mostrada se basan en la misma matriz de nodos, con valores de k=3 y k=4, y con 5.000 iteraciones y número de Jobs=2.
Nota: El uso de las librerías aquí mencionadas mejora drásticamente el rendimiento agrupando los clusters en menos de 1 segundo.

Para cada línea, vamos a cotejar los resultados obtenidos, con la media, varianza, y desviación del nodo (que nos da un reflejo de su funcionamiento).

Aunque K-Means y PCA, parecen tener objetivos muy diferentes, hay una gran interconexión tal como se explica [5],
PCA busca representar todos los vectores de datos n:n como combinaciones lineales de un pequeño número de vectores propios, y lo hace para minimizar el error de reconstrucción cuadrático medio.

K-Means busca representar todos los vectores de datos n:n a través de un pequeño número de centroides de clúster, es decir, representarlos como combinaciones lineales de un pequeño número de vectores centroides de clúster .
PCA se utiliza para la reducción de dimensionalidad / selección de características, cuando el espacio de funciones contiene demasiadas funciones irrelevantes o redundantes o que no podemos utilizar. El objetivo es encontrar la dimensionalidad intrínseca de los datos.

Aquí hay un ejemplo bidimensional que se puede generalizar a espacios dimensionales superiores. El conjunto de datos tiene dos funciones, xx y yy, cada círculo es un punto de datos. Reducimos v1 y v2 a v1 por tener un mayor valor que v2 (realizando la reducción) y utilizando v1 como característica principal y reduciendo la dimensión y los valores a utilizar.

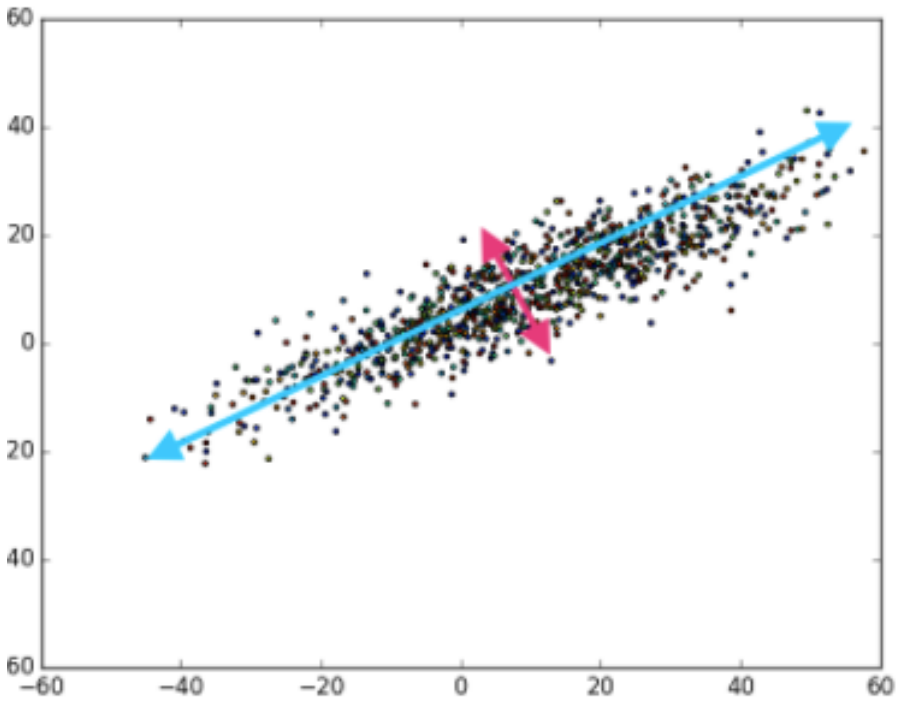


Figure 13 Ejemplo de reducción con PCA

Nuestras librerías de K-Means utilizan la reducción SVD (Single Valued Decompostion):

Para k=3, Iteraciones 5000 y centroides inicialmente aleatorios:

Obtenemos los siguientes valores:

				K=3	K=4	K=5
nodo	media	varianza	Desviación	cluster		
100019206	0.64	0.0378	0.1946	1	1	0
100021662	0.961	0.0005	0.0229	1	2	2
100052112	1.0	0.0002	0.0154	1	2	2
100055671	0.3515	0.0003	0.0173	2	0	1
100066658	0.781	0.0459	0.2144	1	1	0
100067629	0.0	0.0537	0.2317	0	3	4
100075173	0.308	0.0007	0.0267	2	0	1
100117943	0.0	0.0	0.0	0	3	4
100203236	0.2195	0.0056	0.0753	2	0	1
100205414	0.0915	0.0128	0.1133	2	3	4
100208668	0.3505	0.0048	0.0695	0	0	1
100224447	0.779	0.0009	0.0309	2	1	0
100233682	0.933	0.0051	0.0715	1	2	2
100252332	0.6235	0.0311	0.1765	1	1	0
100265331	1.0	0.0003	0.0199	1	2	2
100272294	0.478	0.0705	0.2656	2	1	0
100284574	0.978	0.0006	0.0251	1	2	2
100309685	0.24	0.0090	0.0953	2	0	1
100317715	1.0	0.0017	0.0412	1	2	2
100327214	0.0	0.0	0.0	0	3	4
100355392	0.689	0.0008	0.0294	1	1	0
100389873	0.121	0.0419	0.2048	2	0	1
100408768	0.0325	0.0151	0.1231	0	3	4
100413314	0.2155	0.0090	0.0952	2	0	1
100422568	0.0	0.0001	0.0128	0	3	4

100422926	1.0	0.2111	0.4594	1	1	3
100423635	0.167	0.0581	0.2410	2	0	1
100427449	0.0	0.0208	0.1443	0	3	4

Tabla 14 Listado de nodos, medias, varianza y desviaciones

Logs:

```
transformada [[ -1.75336111e+00  -5.30422588e-01]
 [ -6.41145778e+00   4.57006871e-01] [ -6.85488188e+00   4.35013311e-01]
 [  1.41035852e+00   1.59737742e-01] [ -3.21825316e+00  -1.59409936e-01]
 [  4.49846398e+00  -6.42320722e-01] [  2.10355685e+00   2.34494619e-01]
 [  5.98727694e+00   1.11967931e-01] [  3.15469047e+00   2.13688663e-01]
 [  4.53672402e+00  -3.29355109e-01] [  1.52139403e+00   8.63896955e-02]
 [ -4.12363378e+00   4.36832696e-01] [ -5.88784237e+00   2.64369684e-01]
 [ -2.47635696e+00   1.31356249e+00] [ -6.87371944e+00   3.95197394e-01]
 [ -4.05487068e-01   7.71633620e-01] [ -6.52897769e+00   4.59488971e-01]
 [  2.93755492e+00   3.04918810e-01] [ -6.54480069e+00   9.01092969e-02]
 [  5.98727694e+00   1.11967931e-01] [ -3.03092013e+00   2.54273001e-01]
 [  2.88097261e+00   2.73930498e-01] [  4.93877479e+00   7.12412010e-01]
 [  3.02852282e+00  -4.49703053e-03] [  5.92735593e+00   8.08826239e-02]
 [ -2.65789390e+00  -5.59274034e+00] [  2.95549418e+00   5.09818926e-01]
 [  4.89916896e+00  -4.18951064e-01]]

Clusters:  [1 1 1 2 1 0 2 0 2 0 2 1 1 1 1 2 1 2 1 0 1 2 0 2 0 1 2 0]

Centroides
[[ 5.25357737 -0.05334234] [-4.69684157 -0.18139326]
 [ 2.1763397   0.28334617]]

Inertia[12]: 77.947
```

Observamos que la media +- desviación típica: nos indica:

- Clúster 0: Valores muy bajos o 0.
- Clúster 1: Valores muy altos o 1
- Clúster 2: Valores intermedios.

[12] Inertia: Cuanto menor es más coherente es el cluster. $\sum_{i=0}^{n} min_{\mu \in C} (\|x_j - \mu_j\|)^2$

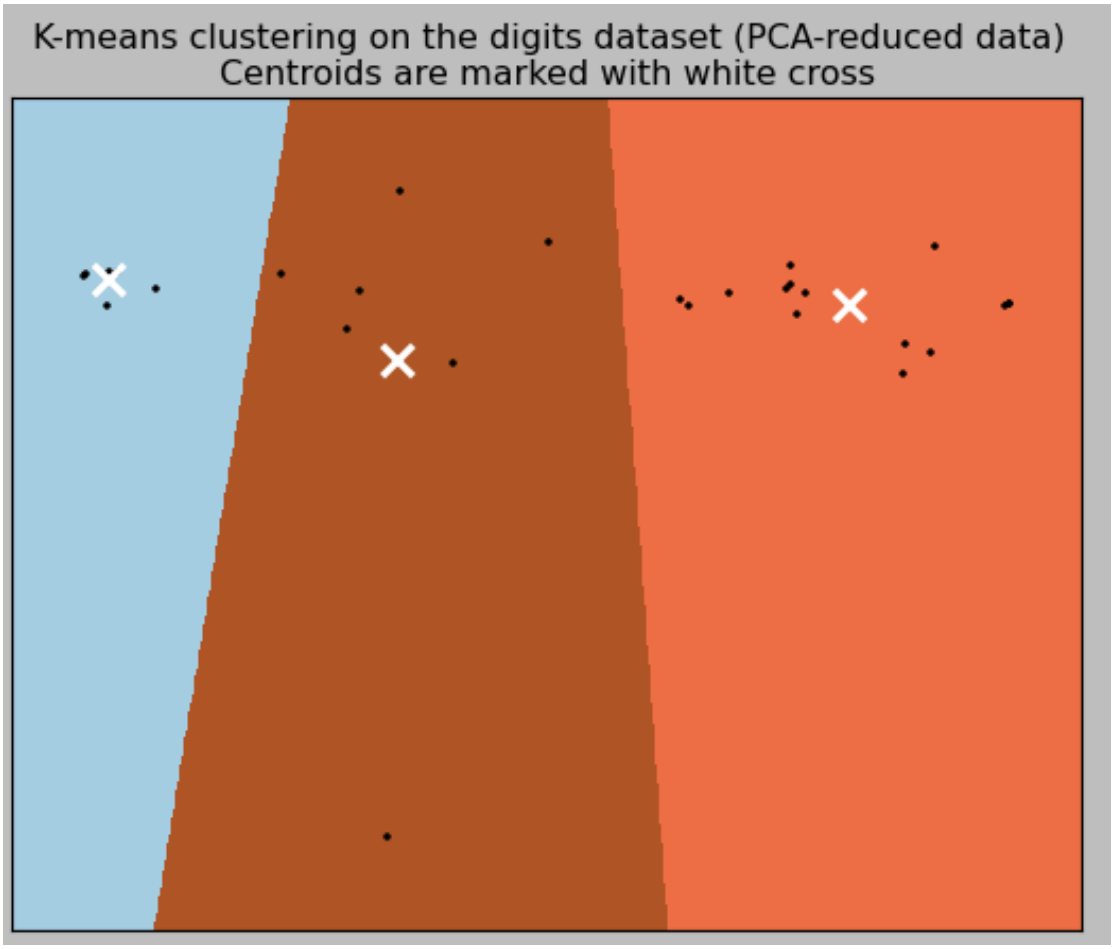


Figure 14 Estructura de los clusters y centroides k=3

Vemos que en este caso el perfilado es correcto.

Para k=4, Iteraciones 5.000, y centroides inicialmente aleatorios

```
Clusters:  [1 2 2 0 1 3 0 3 0 3 0 1 2 1 2 1 2 0 2 3 1 0 3 0 3 1 0 3]
Centroides [[ 2.49906805  0.22231024]
 [-2.52370087 -0.50089586]
 [-6.51694664  0.35019759]
 [ 5.25357737 -0.05334234]]
Distancia Inertia: 49.3379892552
```

Obtenemos:

- Clúster 3, valores a 0 o muy cerca de 0.
- Clúster 2: valores muy cercanos a la media +- desviación con valor 1.
- Clúster 1, valores de media +- desviación, valores intermedios altos

- Clúster 2: valores medio bajos de media+- desviación típica.

El perfilado es correcto

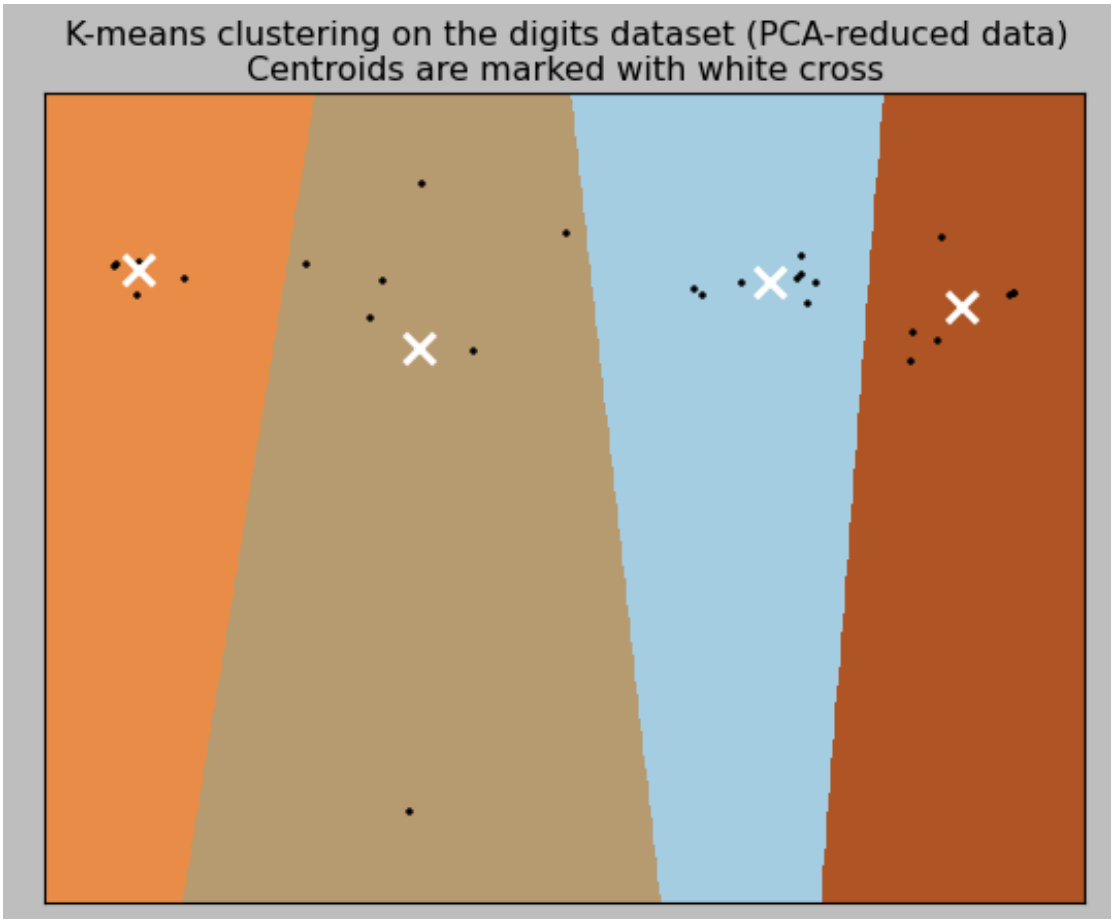


Figure 15 Estructura de clusters y centroides k=4

Para k=5, Iteraciones 5.000 y centroides inicialmente aleatorios

```
Clusters:  [0 2 2 1 0 4 1 4 1 4 1 0 2 0 2 0 2 1 2 4 0 1 4 1 4 3 1 4]
Centroides [[-2.50133537  0.34774488]
 [ 2.49906805  0.22231024]
 [-6.51694664  0.35019759]
 [-2.6578939  -5.59274034]
 [ 5.25357737 -0.05334234]]
Inertia: 19.06895335
```

Vemos que:

- Clúster 4, es 0 o muy cerca de 0.
- Clúster 1, valores de media entre 0,1 y 0.3
- Clúster 2: Valores a 1 o muy cerca del 1.
- Clúster 3, valores intermedios entre el cluster 1 y el cluster 2.

El perfilado es correcto.

NUMERO DE CLUSTERS OPTIMO

Nos queda determinar cual debería ser el valor de k correcto. Aunque no exista un criterio objetivo para la selección del número de Clusters, si que se han implementado diferentes métodos que nos ayudan a elegir un número apropiado de Clusters para agrupar los datos; como son, el método del codo (elbow method), el criterio de Calinsky, el Affinity Propagation (AP), el Gap (también con su versión estadística), Dendrogramas,

Método del codo (Elbow Method)

Este método utiliza los valores de la inercia obtenidos tras aplicar el KMeans a diferente número de Clusters (desde 1 a N Clusters), siendo la inercia la suma de las distancias al cuadrado de cada objeto del Cluster a su centroide:

$$Inertia = \sum_{i=0}^{N} ||xi - \mu||^2$$

Una vez obtenidos los valores de la inercia tras aplicar el KMeans de 1 a N Clusters, representamos en una gráfica lineal la inercia respecto del número de Clusters. En esta gráfica se debería de apreciar un cambio brusco en la evolución de la inercia, teniendo la línea representada una forma similar a la de un brazo y su codo.

El punto en el que se observa ese cambio brusco en la inercia nos dirá el número óptimo de Clusters a seleccionar para ese data set; o dicho de otra

manera: el punto que representaría al codo del brazo será el número óptimo de Clusters para ese data set.

De los datos obtenido de nuestros scripts obtenemos los valores de inertia:

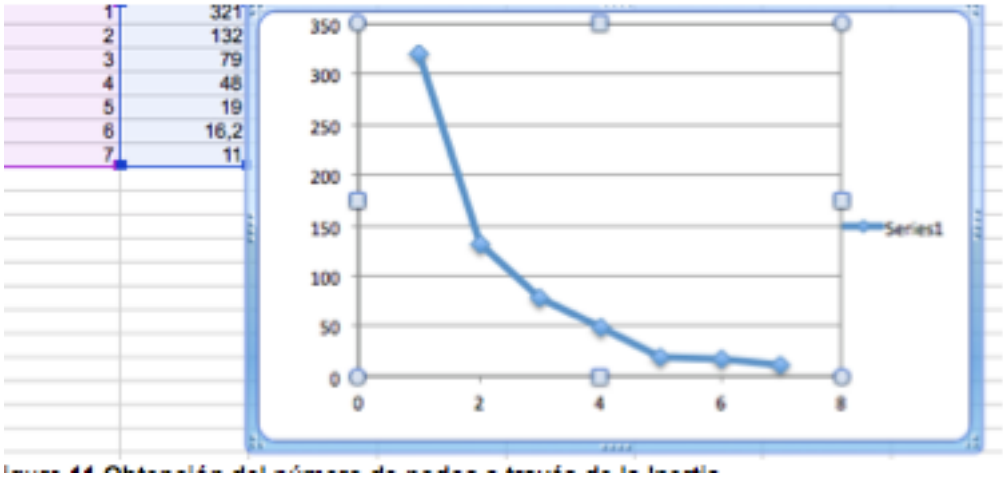


Figure 16 Obtención del número de nodos a través de la Inertia

El valor de clusters debería ser 3.

DB-SCAN

Partiendo de los mismos datos de partida, vamos a verificar cual es el número de cluster “k” adecuado.
Para lo cual vamos a usar el coeficiente de Silhoutte[13], (valor comprendido entre los rangos [-1, 1], que nos indica el correcto agrupamiento cuanto más cercano se encuentre de 1.

Con el siguiente script de Python `db-scan.py` vamos a calcular los clusters basándonos en le agrupamiento por densidad DB-SCAN 14.

Valores de entrada :

- X: Vectores de %avgerage.
- Desviación típica máxima de cada clúster: seleccionamos 0,15 en base a los datos de Tabla 14 Listado de nodos, medias, varianza y desviaciones.
- Eps: valor de la distancia máximo para encontrar al vecino=0,12
- Número de features=168
- Métrica para hallar la distancia: euclidean

Obteniendo los siguientes resultados:

```
Número óptimo de clúster: 4
V-measure: 0,967
Sihoutette coefficient: 0.709
```

[13] Coeficiente de Silhoutte: $SC = \frac{1}{n}\sum_1^n s(x)$ con $s(x) = \frac{b(x)-a(x)}{\max\{a(x),b(x)}$

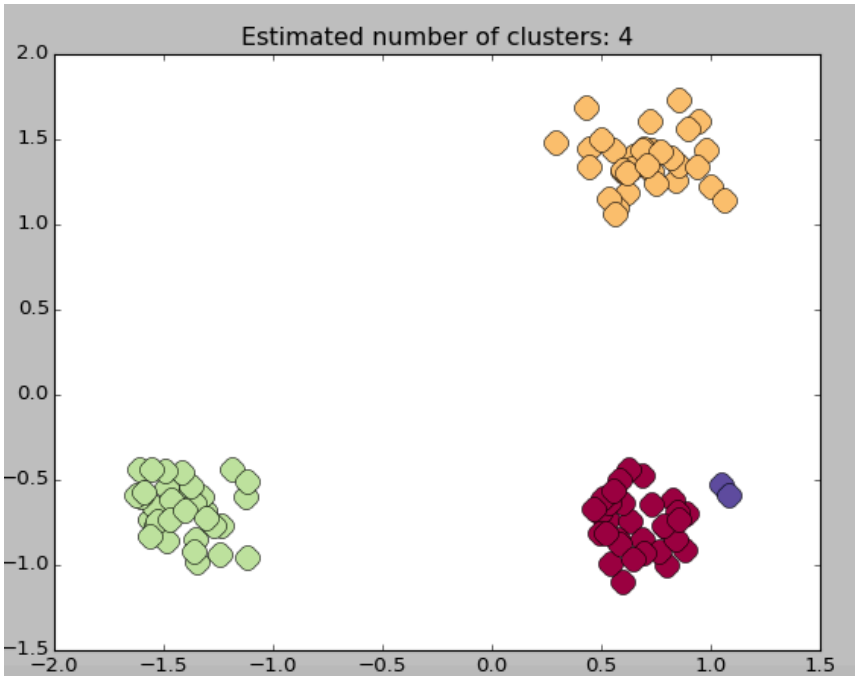


Figure 17 Db-scan clusters, eps=0,12

Encontrando el valor adecuado del coeficiente de Sihoutette en 0,2

```
Homogeneity: 1.000
Completeness: 1.000
V-measure: 1.000
Silhouette Coefficient: 0.888
Estimated number of clusters: 3
```

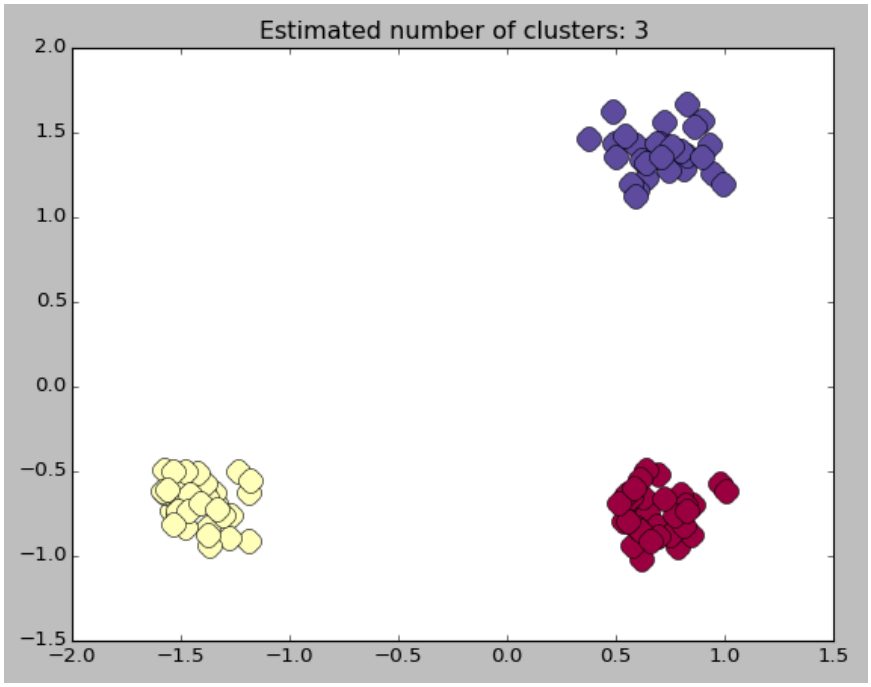


Figure 18 Db-scan número de cluster para eps=0,2

VERIFICACIÓN

VERIFICACIÓN PARA EL PROCEDIMIENTO DE PERFILADO GRUESO

1. Se pretende comprobar que uno o varios nodos se comporta similar a otro (distancia de Hamming pequeña o nula). Para lo cual, el se crearán plantillas de similitud, y se parsearán o pre-procesarán los nodos para disponer del fichero de vectores binarios de uptime con el formato adecuado y serán incluidos en el algoritmo.
   De forma natural la distancia de Hamming de ambos será 0.

VERIFICACIÓN PARA EL PROCEDIMIENTO DE PERFILADO CON K-MEANS

1. Se pretende comprobar que uno o varios nodos se comporta similar a otro (que está comprendido dentro de un cluster. Para lo cual, el se crearán los clusters, y se parsearán o pre-procesarán los nodos para disponer del fichero de %average de uptime con el formato adecuado y serán incluidos en el algoritmo.
   Se espera que el nuevo nodo se encuentre dentro del mismo cluster que el nodo referencia.

   Nota: Al incluir varios nodos de un comportamiento distinto, la generación de los clusters (dinámicos) pueden variar. Esto no debiera ser problema por el alto volumen de datos de partida que procesamos.

   Las resultados obtenidos han sido acordes al planteamiento, funcionando acorde a lo esperado.

Resultados:

Número de líneas de logs	Número de nodos comprendidos	Tiempos de resolución de K-MEANS (segundos)	Tiempo de resolución metauheurístico propio (segundos)
1.000	4	**-**	-
10.000	30	**0.8**	0,025
100.000	118	**1**	0,6
1.000.000	1.093	**28**	11

Tabla 15 Resumen de los tiempos de resolución del problema

Nota: Los datos del rendimiento del meta heurístico fueron descritos en Tabla 10 Comparaciones y tiempos de computación y el número de nodos en Tabla 4 Tamaños de las muestras disponibles

CONCLUSIONES

Para el caso de lo que hemos definido como "**procedimiento propio",** su simplificación a un array de bit y su modelo comparativo:

- En primer lugar perfilado basado en su actividad (números de 1) que dispone el array de bits.
- En segunda instancia distancia comparativa a los demás nodos dentro de su perfil, hace que no le afecten los cambios de forma en la actividad de los nodos.

Se puede asemejar por su forma de trabajo a un cluster jerárquico.
Varía en que no agrupamos los nodos y comparamos el resultado con los demás. Esto nos hace tener comparativas de todos los elementos con todos los elementos, sin embargo para 10.000 líneas de logs procesadas en un sistema mono-hilo esta tarea tarda menos de 2 segundos.
La modificación y adecuación de esta tarea a un cluster jerárquico implicaría poco esfuerzo, sin embargo no creo que aportase mejora.

En el caso de la distribución de K-Means,:
Como se ha mostrado el comportamiento del algoritmo con K-Means utilizando todo el vector, y realizando la transformada, nos permite identificar los cluster y perfilar un nuevo nodo dentro del mismo tras identificar su comportamiento, Su rendimiento es adecuado perfilando 10.000 líneas de los logs en menos de 2 segundos.
Por lo cual se puede usar la solución con un mes (array de 512 elementos) siempre que se definan como feautres.

PROXIMOS PASOS:

Se plantean dos ámbitos de mejora del análisis:
En primer lugar analizar si el modelo de k-mode mejora la clasificación al elegir el elemento central para el cálculo de distancias, a priori se estima que no para valores aleatorios.
Analizar si otro el modelo probabilístico dinámico tal como se ha planteado en 23, es aplicable (a priori sí).

En segundo lugar, mejorar la gestión de acceso a los logs:
En este apartado se pretende dar una alternativa a las limitaciones encontradas de cara a poder procesar ingentes cantidades de datos en la capacidad de almacenar la información, la calidad y estandarización de los procedimientos y velocidad que nos aportan los recursos u orígenes de la información.
Definir cual son las tecnologías actuales para el perfilado de clusters y maching learning implantadas en la industria y los elementos que se utilizan en estas herramientas, para poder definir una mejor solución.
Históricamente en los sistemas los logs han almacenado en texto plano, y posteriormente de forma histórica en BBDD por su disposición a generar datos indexados y vistas con la información pre-procesada. Realizando esta tarea a nivel de recurso y eliminando esta complejidad en el desarrollo de los algoritmos.
Seti@home nos propone estas dos características (18GB en texto plano y 21 GB para la BBDD de mysql).
Ha habido avances de centralización de los mismos y recolección de estas evidencias en entornos distribuidos y con distintos orígenes de información (mezcla de logs en texto plano, BBDD) como la herramienta de graylog e incluso otras opciones como la inclusión de datos no estructurados.
Sin embargo la mayor limitación que podemos encontrar son las consultas a la información.

Tras la incursión de las bases de datos noSQL, se eliminan parte de las deficiencias y limitaciones en los que algunos algoritmos estaban etiquetados y que se han analizado en este estudio.
Hemos de definir sus características de las bases de datos noSQL, ya que a diferencia de las SQL, no todas comparten el mismo lenguaje y no todas trabajan del mismo modo.
Las hay que muestran los datos en un grafo y las hay que lo hacen en documento.

Las bases de datos NoSQL que más se utilizan en entornos productivos de gran carga en procesamiento son:

Tabla 16 Beneficio y contras en las distintas versiones de BBDD noSQL

Nombre	Descripción	Pros	Contras
Neo4J	Base de datos que muestra los datos en un grafo.	Rapidez en las búsquedas (la más rápida). Lenguaje sencillo (Cipher, el más intuitivo y fácil de aprender). App web intuitiva. Api sencilla de utilizar. Licencia GNU (Copyleft).	
OrientDB	Base de datos que muestra los datos en un grafo, forma documento y key value (la más completa de todas)	Rapidez en las búsquedas. Lenguaje. sencillo (Cipher) App web intuitiva. API muy sencilla de utilizar.	Consola web poco consistente. Distinción entre versiones community y profesional.
MongoDB	Base de datos que muestra los datos en documento	Rapidez en las búsquedas. API sencilla de utilizar.	Las búsquedas se ven condicionadas por el modo en que se almacenan la información en la BBDD.. Restringida a licencia.
Cassandra	Base de datos key	Rapidez en las búsquedas.	Configuración

	value	Licencia Apache Software Fundation	

COMPARATIVA DE BUSQUEDA SQL VS noSQL

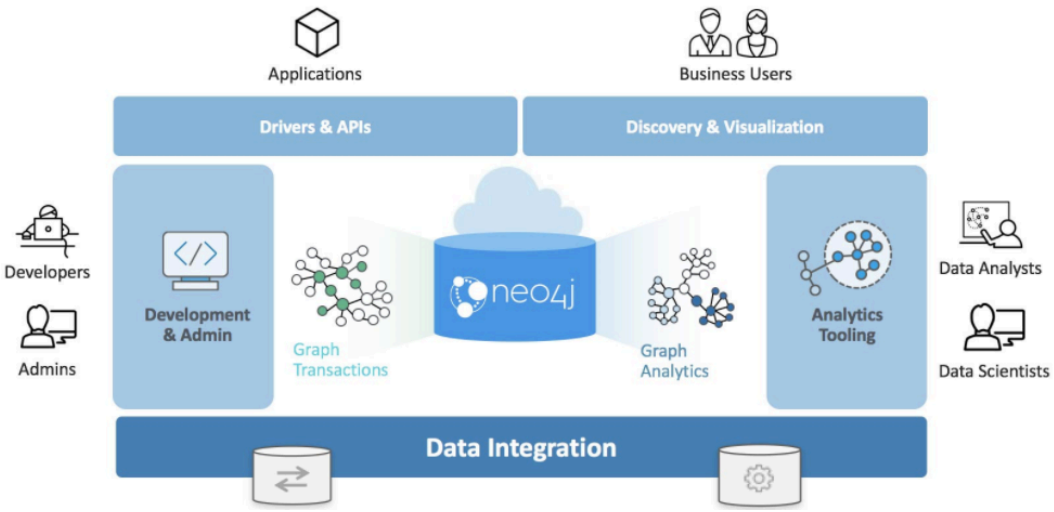
En la siguiente tabla se muestran los tiempos de respuesta entre realizar una búsqueda en una BBDD relacional y una no relacional (Neo4J en particular):

Nivel Profund	Relacional Tiempo (seg)	Neo4j Tiempo (seg)	Registros Devueltos
2	0.016	0.01	≤ 2.500
3	30.267	0.168	≤ 110.000
4	1543.505	1.359	≤ 600.000
5	No Termina	2.132	≤ 800.000

*Partner y Vukotic's Experiment - Graph Databasesby Ian Robinson, Jim Webber, and Emil Eifrem

Figure 19 Comparativa de rendimiento SQL NoSQL

Como se puede apreciar, con un número pequeño de búsquedas los tiempos son casi idénticos, pero al realizar una búsqueda que requiera un gran volumen de datos a devolver, los tiempos son completamente diferentes.



BASES DE LA SELECCIÓN DE ESTA SOLUCIÓN

La solución debe disponer de las siguientes características:

- Tiempo de respuestas altos.
- Use una lógica que permita realizar cambios en el código sin dificultad.
- Disponga una API que permita un uso sencillo.
- Limitación de dependencias. Por ejemplo, al eliminar un dato en una solución SQL, ha de eliminarse de todas las tablas donde se pueda encontrar y/o las referencias. Las bases de datos NoSQL no tienen esta limitación, ya que no existe el join.

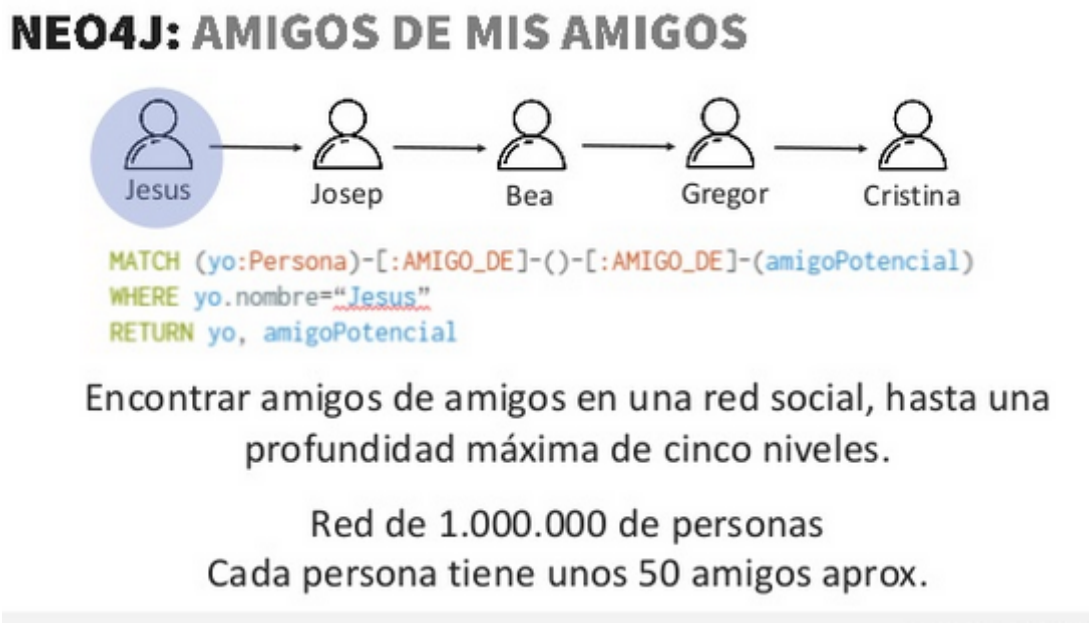
Se propone una base de datos NoSQL de Neo4J (como alternativa de almacenar los logs), por no disponer de las limitaciones anteriormente descritas, disponer de un motor de persistencia y una consola web amigable para su gestión. Permite usar una lógica a nivel de arquitectura sencilla, dispone de una API de fácil integración con tiempos de aprendizaje cortos. El lenguaje de consulta cipher, es muy extendido y sencillo de aprender y utilizar.

Dispone de un licenciamiento AGPLv3 (GNU) que está dentro del marco de Copyleft.

Su última versión estable es la 3.3.0 de Octubre del 2017.

Ejemplo práctico de uso de cipher

Supongamos que tenemos una BBDD con usuarios que son amigos de otros usuarios, y queremos encontrar a los amigos de los amigos de los amigos… Esta profundidad de búsqueda tiene un límite, pero no influye apenas en el tiempo de búsqueda:

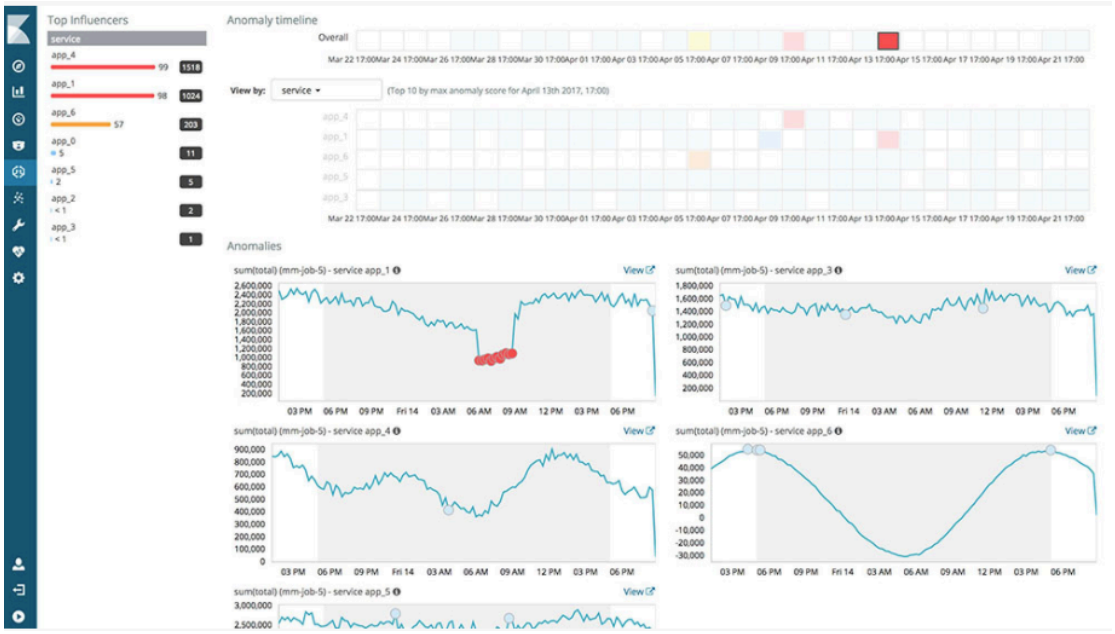


Ahora supongamos que tenemos a la familia Simpson en la BBDD y queremos que nos muestre a todos ellos en un grafo con su relación. El resultado sería el siguiente:

SOLUCIONES COMERCIALES QUE APOYAN LA TENDENCIA

Herramientas comerciales para la clusterización o perfilado de datos masivos: https://www.elastic.co

Dispone de varias soluciones o componentes, que unidos forman una solución de matching Learning para la gestión de patrones tanto de actividad de las aplicaciones, como el perfilado de tendencias.



Componentes:

- Elasticsearch (motor de búsqueda)
- Kibana (Dashboard)
- Beats (agentes para el paso de datos a Elasticsearch o Logstash)

POSIBLE IMPLEMENTACIÓN PROPIA

Estas tecnologías se basan en poder definir nuestra propia solución, basándonos en los componentes que utilizan las herramientas comerciales, y asociadas a la explotación masiva de logs para millones de entradas_

Componente Elasticsearch:

ElasticSeacrh es un motor de búsqueda, que sirven para indexar entornos distribuidos, orientado para grandes volúmenes de datos (escalabilidad horizontal y vertical) y que nos ofrece un acceso en tiempo real y orientado a documentos (JSON) y que no trabaja con schemas de base de datos (estructuras fijas) buscando por patrones
Será un complemento para la solución.

Componente Logstash:

Logstash, herramienta que funciona bajo un JVM que nos permitirá administrar los logs. Nos aporta la posibilidad de usar codecs (permiten cambiar el formato de entrada a otro formato, por ejemplo los datos no son de texto o cambio de estructura). Dispone conectores de entrada y de salida (para almacenar los logs parseado por los codecs).
Y filtros (programados en Grog, que nos ayuda a parsear o filtrar las entradas.

Componente Kibana:

Es un dahboard que nos facilitará la gestión de la información de elastic search.

Neo4j se integra directamente con Logstash, elasticsearch tal como indica su propia web, así como con Python, para poder usar las utilidades de stickit.

GLOSARIO

B

BOINC: (Berkely Open Infrastructure for Network Computing) es un sistema open source liberado bajo la licencia LGPL que permite la computación en grild voluntaria y permite a los investigadores acceder a abundantes recursos de procesamiento obtenidos de la suma de dispositivos personales en todo el mundo.

C

CF: Cluster Features, o características del cluster.

COMPUTACION EN GRID: es un sistema para la compartición de recursos de procesamiento y almacenamiento de varios servidores con un alto rendimiento y bajo demanda. No se trata de varios equipos en paralelo, sino de un equipo virtual que va balanceando recursos libres de muchos servidores para ofrecernos un conjunto escalable de recursos conforme van siendo necesarios. De esa manera, aprovechando recursos disponibles de entre los recursos ofertados, para montar una potente infraestructura virtual distribuida en varios equipos físicos. El concepto de grid no se debe asociar a computación en la nube.

D

DBSCAN: Density Based Spatial Cluster Applications with Noise (KDD 1996)

DHT: (Distributed Hash Table) son un tipo de tablas de hash, almacenan pares (clave, valor) y permiten consultar el valor asociado a una clave, en las que los datos se almacenan de forma distribuida en una serie de nodos (sistemas distribuidos) y proveen un servicio eficiente de búsqueda que permite encontrar el valor asociado a una clave. Para esto último usan un sistema de enrutado que permite encontrar de forma eficiente el nodo en el cual está almacenada la información que se necesita.

DENCURE: Density based Clustering (Hinnedburg & Keim, KDD 1998).

F

FAR: Falsa aceptación.

FDP: Función de densidad probabilística.

FRR : Tasas de Falso rechazo.

O

OPTICS: Ordering Points To Identifiy The Cluster Structure (Ankerst et al SIGMOD 1999)

P

PCA: Principal Component Analysis

PODS: son servidores autónomos en los que los usuarios pueden registrarse y comunicarse con toda la red de microblogging.

RECURSOS:

[1] Setti@Home, Website. (https://setiathome.berkeley.edu/)

[2] BOINC, Website. (https://juncotic.com/boinc-computacion-distribuida-voluntaria/)

[3] Scikit Learn: http://sckit-learn.org

REFERENCIAS

Referencias base:

- [1] Briony J Oates: Researching Information Systems and Computing, 2006, ISBN 978-1-4129-0223-6

Otras referencias:

- [2] Charu C. Aggarwal & Chandan K. Reddy (editors): Data Clustering: Algorithms and Applications. Chapman & Hall / CRC Press, 2014. ISBN 1466558210
- [3] Charu C. Aggarwal Aggarwal & Jiawei Han(editors editors): Frequent Frequent Pattern Pattern Mining. Springer Springer, 2014. , 2014. ISBN 3319078208.
- [4] Ding, C. He. X. [2004], K-Means Clustering via Principal Component Analysis.
- [5] Fisher, D., Knowledge acquisition via incremental conceptual clustering, Machine Learning, VOL. 2, 139-172, 1987.
- [6] Garre, M., Cuadrado, J, Sicilia, M.A, [artículo]."Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software".
- [7] Gutiérrez R., González A., Torres, F. J.A. Gallardo (1994).,

[artículo], "Técnicas de Análisis de datos Multivariable. Tratamiento computacional". Universidad de Granada.

- [8] Jain A.K., Murty M. N., and Flynn P. J. Data clustering: A review. ACM Computing Survey, 31(3):264–323, 1999.
- [9] Lloyd S.P.: Least squares quantization in PCM. Bell Telephone Laboratories, 1957. Published much later in IEEE Transactions on Information Theory 28 (2): 129–137, 1982. DOI 10.1109/TIT .1982.1056489
- [10] MacQueen J. B.: Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1, pp. 281–297. University of California Press, 1967. MR 0214227. Zbl 0214.46201
- [11] Villagra A., Guzmán A. , Pandolfi D. , [artículo], "Análisis de medidas no-supervisadas de calidad en clusters obtenidos por K-means y Particle Swarm Optimization".Universidad Nacional de San Luis.
- [12], Lazaro D., [tesis], "A Middleware for Service Deployment in Contributory Computing System". 2004