Cave Canem: Monitorización de sistemas y detección de intrusiones en PASITO con DDS

Fernando García Aranda, Juan M. Lopez-Soler

Organización

- Introducción
- Objetivo y Requisitos
- Middleware Data Distribution Service
- Arquitectura
- Implementación
- Demo

Cave Canem: **Introducción**

- La "computación centralizada" ha sido sustituida por **entornos distribuidos (virtualizados)** → ***cloud computing***
- La supervisión de la "**salud**" del sistema es clave:
  - La monitorización eficiente de los recursos
  - La seguridad es igualmente relevante
- Existen multitud de herramientas para el tratamiento de estos aspectos. Sin embargo:
  - Proporcionan una "**visión parcial**" (IDS, monitorización...)
  - Son **heterogéneas**: diferentes sistemas de configuración, administración, etc
  - Fuertemente **acopladas** (modelo cliente/servidor)
  - No **escalables** (basadas en un servidor centralizado)
- **PASITO** y sus experimentos son un escenario ideal para el despliegue y test de herramientas de monitorización

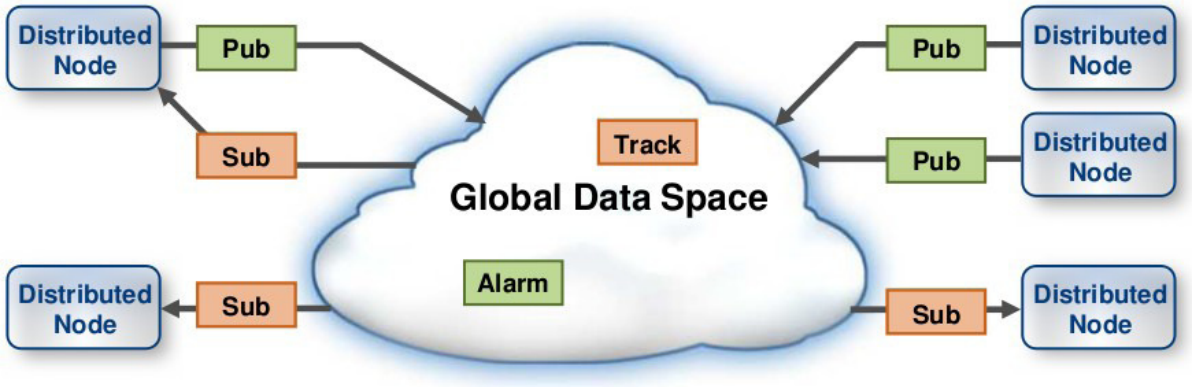
Cave Canem: **Objetivo y Requisitos**

- Diseño e implementación de un sistema para la **supervisión** de sistemas distribuidos

- Requisitos
  - Sea escalable
  - Proporcione una "visión de conjunto"
  - Configuración sencilla y unificada
  - Desacoplado: descubrimiento automático

- Solución propuesta: ***Cave Canem*** plataforma para la monitorización de sistemas y detección de intrusiones en PASITO con DDS

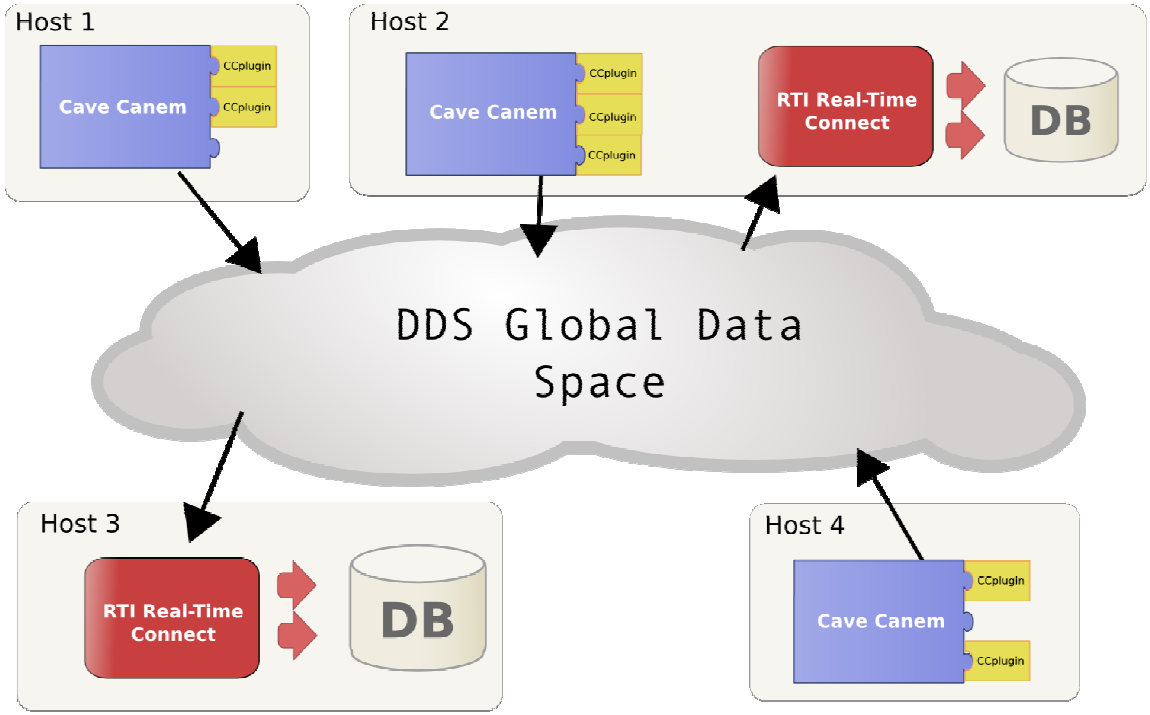
- DDS es un *middleware* estandarizado por la OMG (Object Management Group)
- DDS adopta un modelo *data-centric* siguiendo un paradigma publicación/subscripción
- Conceptos fundamentales:
  - **datawriters,**
  - **datareaders,**
  - **topics**



- Ventajas de DDS:
  - **Flexibilidad** y robustez del modelo *data-centric* (modelo desacoplado)
  - Altas **prestaciones** (real-time) y **escalabilidad**
  - Reduce la **complejidad** de las aplicaciones (perfiles de QoS, filtros)
  - **Interoperabilidad**: independiente del hardware, software y aplicación

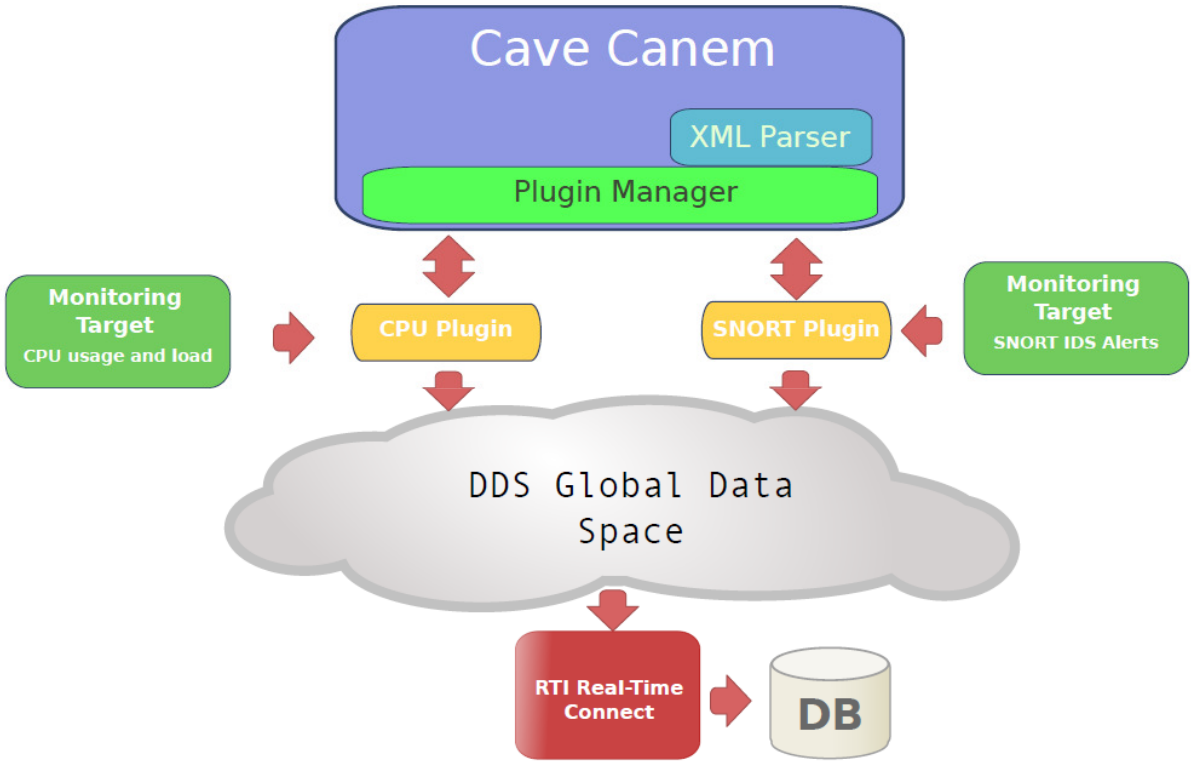
Cave Canem: Arquitectura

- Cave Canem proporciona un "**visión de conjunto**" de la salud del sistema, **combinando** y **normalizando** la información de diferentes tipos de "sensores"
- Se basa en el estándar DDS



Cave Canem: **Arquitectura**

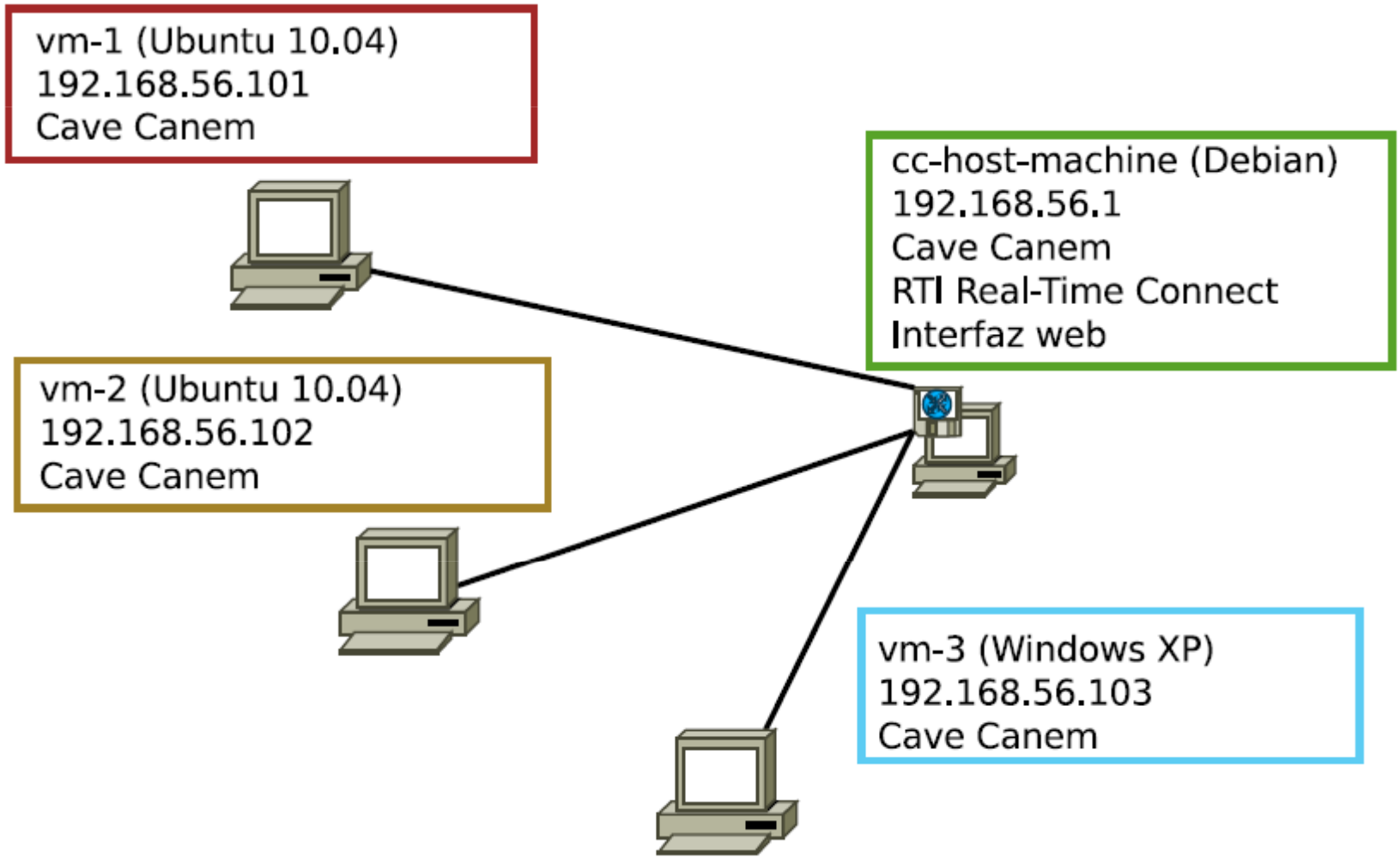
- Cave Canem se basa en una serie de "plugins " que publican la información de interés (*topics*) en el Global Data Space.
- Cave Canem utiliza "RTI real-Time Connect" para integrar DDS con MySQL



Cave Canem: Implementación

- Cave Canem está desarrollado en C++ para Linux y Windows
- Está disponible en https://forja.rediris.es/projects/cusl4-cavecanem/
- Plugins desarrollados:
  - **CPU** (uso de la cpu y carga media)
  - **Memory** (uso de la memoria principal y virtual)
  - **Net Load** (información sobre las interfaces de red del host)
  - **Disk** (información sobre los distintos sistemas de ficheros montados en un host —incluyendo sistemas NFS—)
  - **Proc** (información sobre cada proceso que se está ejecutando en la máquina)
  - **Proc Stat** (Datos generales sobre los procesos que se están ejecutando en el host: número de procesos, número de procesos en cada estado, etc).
  - **Host info** (información sobre el sistema operativo de la máquina y *uptime*)
  - **Snort (**gestión de alertas y detección de intrusiones**)**
- Configuración: se basa en ficheros XML para personalizar las frecuencias de las publicaciones, los perfiles de QoS, los *data types* de los plugin
- Acceso la "visión de conjunto": una interfaz web

Cave Canem: Demo



Contacto: http://tstc.ugr.es/tl

Juan M. Lopez-Soler juanma@ugr.es
Signal Theory, Telematics and Communications Dept.
University of Granada
C/ Periodista Daniel Saucedo Aranda s/n
18071 GRANADA. – SPAIN
phone +34 958 242303