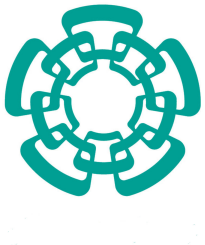


Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Cinvestav Tamaulipas

Método para la supervisión de sistemas de contenedores virtuales basado en multi-modelado

Tesis que presenta:

Fernando Balderas Guzmán

Para obtener el grado de:

Maestro en Ciencias en Ingeniería y Tecnologías Computacionales

Dr. José Luis González Compeán, Co-Director
Dr. Iván López Arévalo, Co-Director

Cd. Victoria, Tamaulipas, México. Septiembre, 2020

Dr. Gregorio Toscano Pulido

Dr. Miguel Morales Sandoval

Dr. Iván López Arévalo, Co-Director

Dr. José Luis González Compeán, Co-Director

Cd. Victoria, Tamaulipas, México., 17 de Septiembre de 2020

A mis padres y hermanas

Agradecimientos

- Agradezco profundamente a mis padres quienes con un gran esfuerzo me han apoyado durante toda mi vida. A mis hermanas Yadira y Paola quienes con su más sincero aprecio me motivan a seguir adelante en mi día a día.
- A mis directores de tesis, el Dr. José Luis González Compeán y el Dr. Iván López Arévalo por brindarme sus conocimientos, por su paciencia y por el tiempo que dedicaron a orientarme en este trabajo de tesis.
- Agradezco a mis revisores de tesis el Dr. Miguel Morales Sandoval y el Dr. Gregorio Toscano Pulido por sus valiosos comentarios que permitir mejorar este trabajo.
- A mis amigos y compañeros de generación, sin excluir a ninguno pero, en especial a Ricardo, Elías y Moguél con quienes también compartí un hogar. Por sus consejos, las experiencias, así como por todos los buenos y malos momentos compartidos. Me llevo un gran recuerdo de ustedes.
- A Giomara, por su apoyo, comprensión y compañía durante esta valiosa etapa de mi vida. Gracias por formar parte de mi vida.
- A todos los investigadores y personal del Cinvestav que de forma directa o indirecta contribuyeron a mi formación personal y científica.
- Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico otorgado, el cual me permitió concentrarme en mis estudios de maestría.
- Al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional por la oportunidad de realizar mis estudios de posgrado.

Índice General

Índice de Figuras

Índice de Tablas

Resumen

Método para la supervisión de sistemas de contenedores virtuales basado en multi-modelado

por

Fernando Balderas Guzmán
Cinvestav Tamaulipas
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2020
Dr. José Luis González Compeán, Co-Director
Dr. Iván López Arévalo, Co-Director

Los contenedores virtuales se han convertido en una solución popular para que las organizaciones solventen problemas de portabilidad, como encapsular una aplicación junto con sus dependencias sin que ésta se vea afectada por el entorno en que se ejecuta. Estos problemas son comunes en sistemas y patrones de software de entrega continua (los cuales deben proveer de manera constante la acción requerida), tales como tuberías de software o flujos de trabajo, ya que éstos comúnmente se ejecutan sobre entornos distribuidos. Garantizar la entrega continua de datos y/o metadatos a cada uno de los componentes de estos patrones y sistemas resulta crucial para este tipo de soluciones, los cuales deben hacer frente a riesgos potenciales que pueden afectar la continuidad del servicio que éstos proporcionan a los usuarios finales. Entre estos riesgos se encuentran afectaciones a propiedades de calidad, tales como eficiencia (que se ve disminuida en escenarios de sobrecarga de algún o algunos componentes), disponibilidad y resistencia a fallas o interrupciones parciales/totales de alguno de sus componentes. Para hacer frente a estos riesgos, es importante identificar los componentes que experimentan sobrecargas de trabajo y aquellos que presentan fallas y/o interrupciones de servicio. En esta tesis se propone solventar las deficiencias de los sistemas y patrones de entrega continua empleando un método de supervisión que permite identificar el punto de falla mediante una representación abstracta. La supervisión consiste en identificar, monitorear e indexar el estado de cada componente del sistema en cuestión, con base en aspectos no funcionales como eficiencia y confiabilidad. Para las tareas de supervisión se aplicó un enfoque de multi-modelado a los

componentes del sistema, el cual consiste en combinar diferentes enfoques de abstracción como secuencial, funcional, estructural, entre otros. Este multi-modelado produce una representación abstracta de cada componente del sistema, lo que permite agilizar las tareas de supervisión. La evaluación experimental reveló la factibilidad de utilizar el método propuesto para la supervisión del estado en sistemas de contenedores virtuales.

Abstract

Multi-modeling based method for supervision of virtual container systems

by

Fernando Balderas Guzmán
Cinvestav Tamaulipas
Center for Research and Advanced Studies, 2020
Dr. José Luis González Compeán, Co-advisor
Dr. Iván López Arévalo, Co-advisor

Virtual containers have become a popular solution for organizations to solve portability problems, such as encapsulating an application along with its dependencies without it being affected by the environment in which it runs. These problems are common in continuous delivery software systems (which must constantly provide the required action) and patterns, such as software pipelines or workflows, as these commonly run over distributed environments. Ensuring the continuous delivery of data and/or metadata to each one of the components of these patterns and systems is crucial, as these kinds of solutions must face potential risks that may affect the continuity of the service that they provide to end-users. Among these risks are effects on quality properties, such as efficiency (which is diminished in overload scenarios of one or some components), availability, and resistance to partial/total failures or interruptions of any of its components. To cope with these risks, it is important to identify those components that experience overloads and those that have failures and/or service interruptions. In this thesis, it is proposed to solve the deficiencies of these kinds of systems by the use of a supervision method capable to identify the point of failure through an abstract representation. Supervision consists of identifying, monitoring, and indexing the status of each component of the system, based on non-functional requirements like efficiency and reliability. For the supervision tasks, a multi-modeling approach was applied to the system components, which consists of combining different abstraction levels such as sequential, functional, structural, among others. This multi-modeling produces an abstract representation of each component of the system,

which allows us to speed up the supervision tasks. The experimental evaluation revealed the feasibility of using the proposed method for condition monitoring in containerized systems.

Nomenclatura

SECI	Sistema de Entrega Continua de Información
ETL	Extracción, Transformación y Carga
P&F	Patrón Filtros y Conectores
WF	Patrón Flujos de Trabajo
FC	Patrón fabricas de procesamiento de software
MV	Máquina Virtual
CV	Contenedor Virtual
DevOps	Software development (Dev) + IT operations (Ops)
LXC	Contenedores Linux
LXD	Demonio de Contenedores Linux
E/S	Entrada(s)/Salida(s)
SOA	Arquitectura Orientada a Servicios
DFP	Programación de Flujo de Datos
OLAP	On-Line Analytical Processing
DFD	Diagrama de Flujo de Datos
APM	Gestión del Rendimiento de Aplicaciones
XML	Extensible Markup Language
XDR	External Data Representation
RRDTool	Round Robin Database Tool
AWS	Amazon Web Services
BB	Building Blocks
WoT	Web of Things
IoT	Internet of Things
IoT-A	Internet of Things Architecture
ARM	Architecture Reference Model
PC	Computador físico
FS	Sistema de archivos
API	Interfaz de Programación de Aplicaciones
REST	Representational state transfer
IDA	The Information Dispersal Algorithm
RAMA	Red Automática de Monitoreo Atmosférico
SIMAT	Sistema de Monitoreo Atmosférico
YAML	YAML Ain't Markup Language (human-readable data-serialization language)
TD	Thing Description
JSON-LD	JavaScript Object Notation for Linked Data

1

Introducción

En este capítulo se presentan los antecedentes y la motivación que dieron origen a esta tesis. A partir de ello se presenta la hipótesis y los objetivos para corroborarla. Para ello se describe de manera general la propuesta de solución, así como la metodología de trabajo seguida.

1.1 Antecedentes y motivación

Desde su creación, las tecnologías de virtualización han impulsado la escalabilidad y flexibilidad que demandan los servicios ofrecidos por el cómputo en la nube, permitiendo crear múltiples servidores virtuales (instancias) sobre un mismo equipo físico. Recientemente estas tecnologías de virtualización también han sido utilizadas para el desarrollo de sistemas de software, debido a que las necesidades de interacción con los usuarios se encuentran en constante evolución.

Los Sistemas de Entrega Continua de Información (SECI) basados en contenedores virtuales se están convirtiendo en una solución cada vez más adoptada por las organizaciones para resolver problemas que se presentan durante la integración de múltiples piezas de software requeridas para

construir sistemas y servicios complejos [7, 8]. Este tipo de soluciones de software permiten construir servicios en forma agnóstica[1], los cuales son esencialmente distribuidos, pero sin restricciones de plataformas, sistemas operativos o lenguajes de programación [24, 54]. Las soluciones agnósticas resuelven problemas de portabilidad en el despliegue en diferentes tipos de infraestructuras tales como servidores, clústeres o nubes (privadas o físicas).

Usualmente estas soluciones son construidas mediante patrones de diseño de software, tales como tuberías de procesamiento, flujos de trabajo y fábricas de procesamiento [3, 17, 24, 25, 32]. Un patrón de diseño de software es una forma de describir diseños recurrentes usados en el desarrollo de soluciones de software. Estos patrones se representan mediante un esquema genérico donde se definen y describen los componentes que lo constituyen, sus responsabilidades, relaciones y las formas en que colaboran [42, 50].

Los patrones más utilizados son los sistemas de tuberías conocido como *filtros y conectores* (Pipes & Filters (P&F), por su acepción en inglés). Estos patrones permiten concatenar aplicaciones (filtros) mediante interfaces de entrada/salida (conectores) creando *tuberías de software*. La Figura 1.1(a) muestra un ejemplo de un patrón basado en filtros y conectores.

Un filtro se basa en el modelo tradicional de procesamiento ETL (extracción, transformación y carga [53]), lo cual significa que un filtro *extrae* contenidos de una fuente de datos, realiza *transformaciones* a los datos entrantes para agregarles algún valor y *entrega* los resultados a un almacén de datos u otra aplicación.

Usualmente se utilizan aplicaciones que aseguran requerimientos no funcionales, tales como confiabilidad (codificación/decodificación) y seguridad y eficacia (en espacio de almacenamiento). Estos requerimientos pueden variar dependiendo del contexto y la función que se le desee aplicar a cada contenido que procesa una tubería de software.

En la práctica, un filtro es usualmente desarrollado como una caja negra que funciona como un sistema autónomo e intercambiable, lo cual permite que los filtros sean organizados de manera

[1] Transparente, sin conocimiento de la infraestructura subyacente.

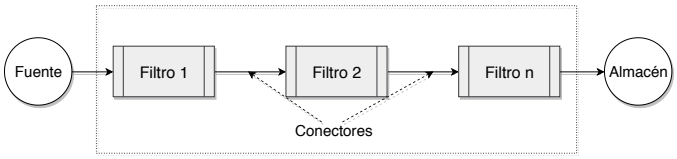
secuencial, enlazándose mediante conexiones llamadas conectores [42, 50]. Esto significa que, al terminar de ejecutarse un filtro, éste invoca la ejecución del siguiente filtro (si lo hubiera) o declara la finalización de la tubería, en el caso de ser el último filtro.

Los flujos de trabajo o Workflows (WF), por su traducción del inglés, son estructuras de procesamiento que implementan los patrones de filtros y conectores, permitiendo gestionar y automatizar el flujo de datos de un proceso, asegurando que éste último siga un orden correcto y garantizando la entrega continua a cada filtro. Este tipo de patrón es utilizado en entornos científicos [12, 20] y tecnológicos [1], donde se procesan grandes cantidades de datos que usualmente requieren una basta infraestructura de recursos. Al unir un conjunto de aplicaciones (comúnmente distribuidas) se pueden crear soluciones de procesamiento colaborativo, permitiendo a las organizaciones involucradas abordar problemas de gran dimensión, mediante el uso de una sola aplicación capaz de procesar una gran cantidad de datos [1, 3, 28, 32]. Los flujos de trabajo se representan como grafos acíclicos dirigidos cuyas estructuras (nodos y aristas) se pueden configurar para definir la manera en la que se ejecutan las tareas, cómo fluye la información y cómo se realiza el cumplimiento de estas tareas. La Figura 1.1(b) muestra un ejemplo de un patrón basado en flujos de trabajo donde se pueden apreciar dos tuberías o flujos distintos. El primero comprende dos filtros encerrados con color azul y el segundo comprende tres filtros encerrados en color verde.

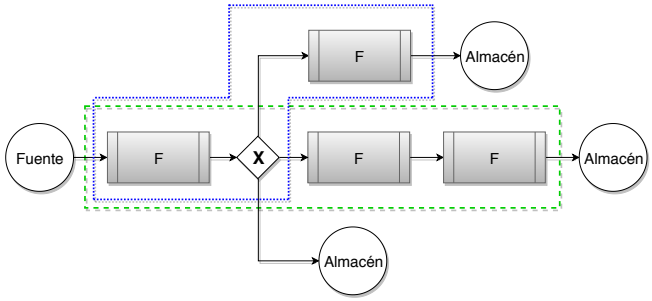
A partir del concepto de WF, las fábricas de procesamiento de software o Fabric Computing (FC), por su acepción en inglés, realizan transformación de datos y configuración de estructuras de WF a través de un paradigma de mallado de nodos interconectados pero débilmente acoplados, permitiendo asignar y reasignar recursos dinámicamente sin depender de la infraestructura sobre la que se ejecutan [47]. Una fábrica de procesamiento tiene las características de ser escalable, eficiente y heterogénea en sus recursos. Usualmente, este tipo de patrones son usados para construir sistemas de hardware; sin embargo, recientemente se han empleado para desplegar arquitecturas de microservicios [13].

La tecnología de microservicios permite a las organizaciones crear un ecosistema de aplicaciones (nodos), que al acoplarse con otros nodos construyen patrones nuevos que representan cada uno

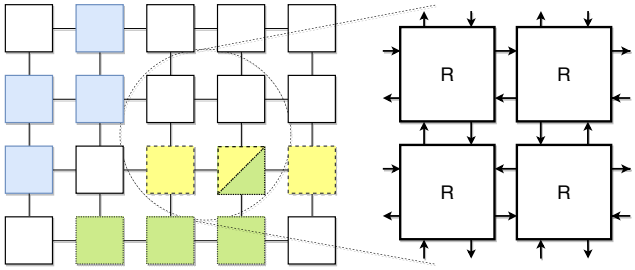
a una solución diferente, la cual puede ser asignada a un cliente o usuario final. La Figura 1.1(c) muestra un ejemplo de un patrón basado en fábricas de procesamiento de software con tres soluciones, cada una emplea un tipo de línea y color diferente. La solución de color azul utiliza cuatro nodos, la solución color amarillo usa tres nodos y la solución color verde utiliza cuatro nodos, cabe aclarar que las últimas dos soluciones coinciden en uno de los nodos. Los recuadros con la R son los recursos con los dispone la solución, estos pueden ser un servicio compuesto de múltiples aplicaciones o una sola aplicación.



(a) Patrón filtros y conectores.



(b) Patrón flujos de trabajo.

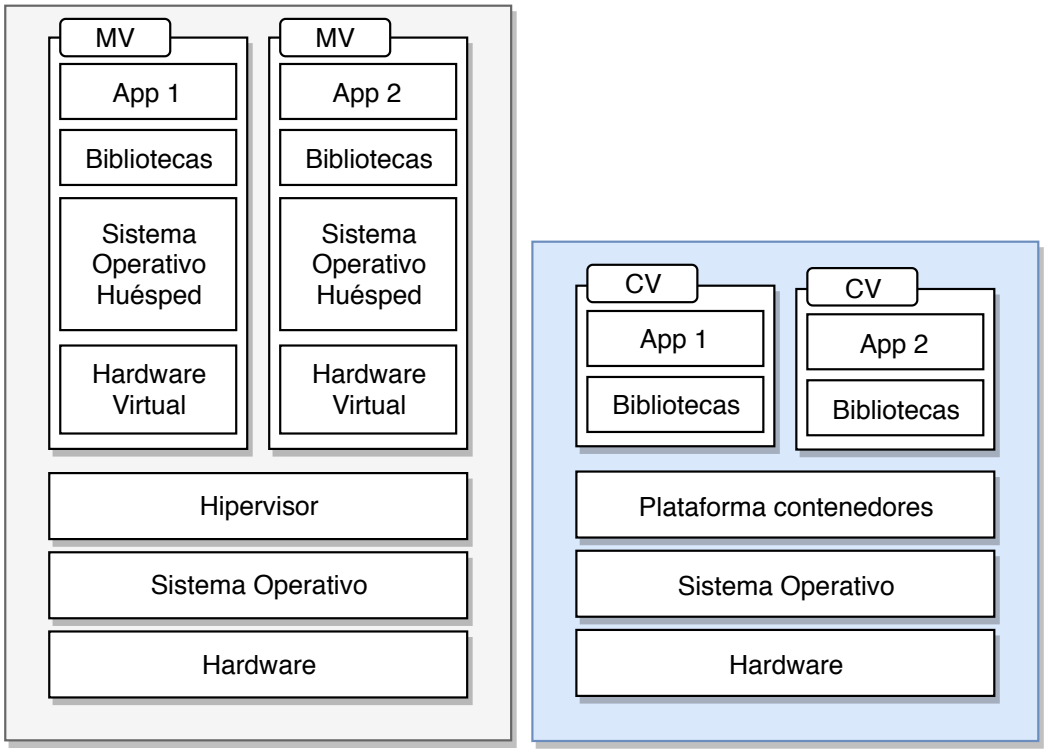


(c) Patrón fábricas de procesamiento.

Figura 1.1: Ejemplos de patrones de diseño de software.

Para el despliegue de este tipo de patrones se pueden utilizar dos enfoques. El primero consiste en realizar el encapsulado de los nodos/filtros (aplicaciones) dentro de máquinas virtuales (MV), las cuales se construyen mediante la virtualización de hardware. Esto significa que cada máquina virtual mantiene reservados los recursos que le fueron asignados incluso cuando ésta no se encuentre ejecutando aplicaciones. Este es un problema que se acentúa al desplegar múltiples máquinas virtuales. El segundo enfoque consiste en encapsular aplicaciones en contenedores virtuales (CV), que a diferencia de las máquinas virtuales, no hacen uso de la virtualización de hardware, por lo que

los recursos se encuentran disponibles bajo demanda para los contenedores activos. La Figura 1.2 muestra la comparativa de los enfoques con máquinas virtuales (Figura 1.2(a)) y con contenedores virtuales (Figura 1.2(b)). La principal diferencia entre ambos enfoques es el uso de un hipervisor en los enfoques con máquinas virtuales, el cual es reemplazado por la plataforma de contenedores en el enfoque con contenedores virtuales. No obstante, independientemente del modelo de despliegue que se elija para crear un patrón de software, una vez desplegado dicho patrón, se requiere que las soluciones basadas en patrones de software provean aspectos cualitativos (no funcionales) a los SECI, tales como P&F, WF y/o FC.



(a) Máquinas virtuales. (b) Contenedores virtuales.

Figura 1.2: Comparativa entre máquinas virtuales y contenedores virtuales.

Considerando los dos enfoques anteriores, optar por el encapsulado en contenedores virtuales suele ser una mejor opción [39], lo cual conduce a la siguiente etapa, que es el representar estas soluciones de software de manera que sean fácilmente entendibles para un usuario.

En la literatura se considera que todo sistema físico o abstracto (en los que se encuentran los SECI) son susceptibles de ser modelados desde la perspectiva de un modelador humano. Con base en el interés del modelador, éste puede emplear un multi-modelado que involucre: i) conceptos ontológicos ii) premisas de representación, iii) tipos de conocimiento y iv) niveles de agregación [9]. Todo WF en esencia es un diagrama de flujos de datos, en consecuencia, un SECI puede modelarse adecuadamente mediante un enfoque de multi-modelado manteniendo una adecuada relación entre las características de P&F, WF y/o FC.

El enfoque de multi-modelado es propicio para emplearse en la representación de un SECI, sin embargo éste requiere de un modelo que pueda adaptarse/trasladarse de manera transparente a cualquier infraestructura que se emplee para el despliegue del mismo. En este sentido, se requiere seguir lineamientos que sean de aceptación común entre diferentes sistemas de cómputo. La Web de las Cosas (Web of Things por su acepción en inglés) plantea una serie de especificaciones que permiten modelar dispositivos (artifacts), tanto físicos como abstractos, de forma que éstos tengan una representación única de aceptación común. Una vez definida esta representación, puede adaptarse y/o expandirse de manera abstracta o física.

1.2 Planteamiento del problema

Los SECI se enfrentan a problemas que resultan críticos en escenarios reales, los cuales se podrían convertir en interrupciones de continuidad. Por ejemplo, se requiere que elementos virtuales (contenedores virtuales o máquinas virtuales) que encapsulan a los componentes de un SECI sean supervisados para verificar que se encuentran realizando sus funciones adecuadamente. Esto resulta crítico debido a las dependencias que crea cada componente dentro de un patrón, como son los múltiples conectores utilizados o la disponibilidad de los nodos interconectados. Por tanto, el estado de "salud" de los conectores y de los nodos que forman parte de un patrón debería ser adecuado para garantizar la entrega continua. En este sentido, dos aspectos no funcionales, confiabilidad y

eficiencia, deben ser supervisados para garantizar que no se interrumpa la entrega continua.

Para garantizar la *confiabilidad* de procesos de entrega continua se debe revisar el estado de los contenedores virtuales que soportan al sistema para asegurar la entrega continua de datos y evitar la interrupción del procesamiento continuado de la información. La *eficiencia* es el segundo requerimiento no funcional que un sistema de entrega continua debe proporcionar a las organizaciones para que los SECI eviten o identifiquen cuellos de botella en sus estructuras.

Los sistemas de supervisión existentes permiten el monitoreo de sistemas completos como los creados con P&F, WF y FC [31, 37]; sin embargo, no diagnostican requerimientos no funcionales (confiabilidad y eficiencia) ni permiten hacer una representación/modelado de estos componentes, sino más bien de sus componentes individuales. Existen casos específicos en los cuales herramientas DevOps [30] para SECI, como Jenkins [1], que ofrecen características remarcables sobre el rendimiento global de un SECI; sin embargo, sólo producen una representación parcial de su estado global. Otros enfoques permiten el monitoreo y gestión de rendimiento (eficiencia) de soluciones de software [19], pero a nivel de computador físico sin tomar en cuenta si el sistema está desarrollado mediante un patrón específico.

1.3 Hipótesis

De la problemática y el marco teórico mostrados en los párrafos anteriores se plantea la siguiente hipótesis de investigación:

La indexación de variables del comportamiento dinámico de propiedades (confiabilidad y eficiencia) de los componentes de un SECI (basado en contenedores virtuales) en sus diferentes etapas permitirá caracterizar a tales componentes, produciendo una representación abstracta mediante la cual se podrá supervisar en forma integral el estado actual del sistema de contenedores virtuales.

1.4 Objetivos

Para cumplir con la hipótesis planteada en este trabajo de investigación se establecen los siguientes objetivos.

Objetivo general

Desarrollar un método para la supervisión dinámica de los componentes de un SECI produciendo una representación abstracta del sistema de contenedores virtuales.

Objetivos particulares

- Desarrollar un método de monitoreo de variables de comportamiento dinámico de un SECI.
- Definir un esquema de estructuras de cubo de datos para gestionar la supervisión de las propiedades de confiabilidad y eficiencia de los componentes de un SECI.
- Definir una representación abstracta del análisis del comportamiento dinámico de las variables de un SECI.

1.5 Solución propuesta

Para resolver la problemática descrita previamente se plantea un método que permita la supervisión de los componentes de un sistema SECI basado en contenedores virtuales. Las etapas de este método se describen a continuación y se representan gráficamente en la Figura 1.3.

- Monitoreo: en esta etapa se realiza una revisión continua de los contenedores virtuales de un SECI. Para ello se utilizan técnicas de paso de mensajes, lo cual permite revisar si los contenedores se encuentran activos y posteriormente describir las métricas de comportamiento de cada uno.

- Indexado: en esta etapa se genera una estructura de hipercubo de datos para almacenar los datos de las métricas obtenidas en la etapa de monitoreo.

- Representación: en esta etapa se caracterizan los componentes utilizando los datos obtenidos en el indexado. Además, mediante un enfoque de multi-modelado, se realiza la representación en múltiples niveles del sistema y sus componentes.

- Diagnóstico y reporte de estado: esta etapa involucra la intervención de un usuario al cual se le presenta un reporte de estado del sistema. Mediante este reporte, el usuario puede identificar el punto de falla, si hubiera, y alguna posible solución, si la hubiere.

Figura 1.3: Etapas del método de solución.

1.6 Metodología de trabajo

En esta tesis se siguió una metodología compuesta por cuatro etapas, las cuales se describen a continuación:

- **Etapa 1: Delimitación del problema**

  En esta etapa se realizó la redacción del protocolo de tesis, el cual se dividió en dos partes, revisión de la literatura y revisión de herramientas de contenedores.

  - **Revisión de la literatura**

    Se realizó una revisión del estado del arte respecto a las propuestas relacionadas con el problema y que lo aborden desde diferentes enfoques, haciendo especial énfasis en los siguientes temas:

    - *Patrones arquitectónicos:* se revisaron los patrones comúnmente utilizados en el desarrollo de sistemas de software, buscando las variaciones y relaciones entre cada uno de ellos.
    - *Sistemas de entrega continua de información:* se investigaron soluciones con las características de un SECI, esto incluyó la forma en que fueron desarrollados y los tipos de patrones que utilizan.
    - *Sistemas de monitoreo:* se buscaron propuestas que realicen algún tipo de monitoreo a componentes o sistemas con el objetivo de encontrar técnicas que permitan realizar el monitoreo a cada uno de los componentes de un SECI.
    - *Técnicas de modelado:* se estudiaron técnicas para representar un sistema junto con sus componentes de una manera fácilmente comprensible para un usuario, buscando formas de abstraer los componentes, sus conexiones y estado funcional.

  - **Revisión de herramientas de contenedores**

    Se revisaron, analizaron y probaron diferentes herramientas que realizaran la creación, despliegue y eliminación de contenedores virtuales.

- **Etapa 2: Diseño y desarrollo del método para la supervisión de sistemas de contenedores virtuales basado en multi-modelado**

Durante esta etapa se desarrolló un método para la supervisión de sistemas de contenedores mediante una representación abstracta, para lo que se realizó una serie de actividades para obtener los módulos que lo conforman. Estos componentes se describen a continuación:

- **Diseño y desarrollo del módulo de monitoreo:** en este módulo se realizó el seguimiento del rendimiento de cada componente de un sistema permitiendo obtener datos representativos de cada uno de ellos.
- **Diseño y desarrollo del módulo de indexado:** en este paso se desarrolló un módulo que guarda de manera periódica las métricas del sistema examinado en estructuras de datos multidimensionales de tipo cuboide, que corresponden a requerimientos no funcionales de cada componente dentro del sistema analizado.
- **Diseño y desarrollo de la representación abstracta:** se realizó la caracterización de los componentes, así como del sistema completo, mediante las métricas del sistema de indexado. Esto permitió obtener una abstracción que facilite a un usuario la comprensión del sistema.
- **Diagnóstico y reporte de estado:** se implemenó un mecanismo que genera un resumen del estado de un sistema obtenido con la representación del paso anterior, el cual sirve para el diagnóstico del mismo.

- **Etapa 3: Implementación de un prototipo funcional**

  En esta etapa se implementó un prototipo siguiendo una arquitectura de clúster para posteriormente realizar la experimentación.

  - **Integración de los componentes de software desarrollados:** Se realizó la integración de los módulos de software correspondientes a la propuesta de solución.

- **Etapa 4: Evaluación experimental**

En esta etapa se emplearon diversos contenedores ya existentes, los cuales se desplegaron montando escenarios para llevar a cabo la evaluación del prototipo. Al prototipo se le realizaron las mejoras que fueron necesarias para solventar todos los escenarios.

- *Metodología de evaluación*: se definió la metodología a utilizar durante la evaluación experimental, la cual está compuesta de dos etapas: la evaluación controlada y un caso de estudio.
- *Definición de métricas de rendimiento:* se definieron las métricas que se podían medir durante la experimentación, mismas que se utilizaron posteriormente en el análisis de resultados. Éstas son los tiempos de respuesta y tiempos de servicio.
- *Despliegue de escenarios de evaluación:* se definieron y desplegaron diferentes escenarios que sirvieron para evaluar el prototipo.
- *Experimentación:* una vez desplegados los escenarios, se realizaron pruebas sobre sistemas desarrollados mediante contenedores virtuales a fin de evaluar la solución propuesta.

- **Etapa 5: Análisis y presentación de resultados**

  - *Análisis de resultados experimentales:* con base en los resultados obtenidos, se realizó el análisis de las métricas elegidas.
  - *Discusión y conclusiones:* se realizó la interpretación de los resultados a fin de corroborar la hipótesis, lo que permitió también elaborar las conclusiones.
  - *Redacción del documento de tesis:* se fue generando paulatinamente (revisando y aumentando el contenido) el documento de tesis corroborando que se describa de manera detallada lo realizado en cada una de las etapas.

1.7 Organización de la tesis

Este documento de tesis se encuentra organizado de la siguiente manera. En el Capítulo 2 se describe el fundamento teórico requerido para el desarrollo de esta tesis. El Capítulo 3 presenta una revisión del estado del arte respecto a trabajos relacionados al monitoreo, desarrollo de sistemas mediante estilos arquitectónicos y la representación de los mismos. En el Capítulo 4 se describe el diseño del método propuesto, así como la implementación de un prototipo y sus componentes. El Capítulo 5 describe la metodología que se siguió para realizar la evaluación experimental. Finalmente, en el Capítulo 6 se muestran las conclusiones obtenidas en este trabajo de tesis.

2

Marco teórico

En este capítulo se presentan los distintos conceptos teóricos y tecnologías involucradas en el desarrollo de este trabajo de investigación. Estos se han dividido en tres secciones. La Sección 2.1 corresponde a los conceptos y tecnologías referentes a los sistemas basados en contenedores virtuales. La Sección 2.2 muestra los conceptos involucrados en la tarea de supervisión de sistemas de contenedores. Finalmente, los conceptos y tecnologías relacionados con la representación de soluciones de software se describen en la Sección 2.3.

2.1 Sistemas basados en contenedores virtuales

Los sistemas basados en contenedores virtuales se refieren a las soluciones de software (aplicaciones) que tienen como base de ejecución los contenedores virtuales. De esta manera aprovechan los beneficios de los contenedores permitiendo que se puedan ejecutar varias aplicaciones de manera paralela y aislada unas con otras. En esta sección se describen algunos temas para comprender mejor estos sistemas.

2.1.1 LXC y LXD

Los contenedores linux (LXC[1]) permiten crear espacios virtuales que incluyen a los procesos y un espacio de red. Éstos utilizan *namespaces* (espacios de nombre) para el aislamiento entre procesos y *cgroups* (grupos de control) del kernel para realizar la gestión de recursos, el cual limita la cantidad de CPU, memoria, E/S de disco y uso de la red para uno o más procesos. Por otra parte, LXD es un demonio que permite controlar los contenedores LXC, enfocándose principalmente en los contenedores de sistema o contenedores de infraestructura. Estos contenedores usualmente tienen un mayor tiempo de vida en periodos activos y parten de una imagen de distribución limpia. Por lo tanto, al usar LXD también se está utilizando LXC. Los componentes principales de LXD son los siguientes:

- Contenedores: es un conjunto de elementos entre los que se encuentran un sistema de archivos, configuraciones (recursos, entorno, seguridad, etc.), un conjunto de perfiles, propiedades (arquitectura de contenedor, efímera o persistente y el nombre) y estado de ejecución.
- Instantáneas: son similares a un contenedor pero estas son inmutables. Se pueden renombrar, eliminar y restaurar, pero no editar. Una característica importante es que permiten guardar el estado de ejecución, por lo que al realizar una recuperación, ésta incluirá el estado de CPU y memoria obtenido al realizar la instantánea.
- Imágenes: un contenedor es creado a partir de una imagen, las cuales son usualmente distribuciones Linux limpias que pueden ser identificadas por su hash sha256.
- Perfiles: es una forma de configurar los contenedores. Se define la configuración y a los contenedores que aplica, ya que un contenedor puede tener varios perfiles aplicados.
- Controles remotos: permiten conectar la línea de comandos cliente con uno o más servidores LXD. Estos servidores pueden ser locales o estar en la red.

[1] https://linuxcontainers.org/

2.1.2 Docker

Docker es un motor de contenedores de código libre que utiliza una virtualización ligera permitiendo automatizar el despliegue de aplicaciones mediante contenedores virtuales. Esto significa que puede encapsular las aplicaciones dentro de un contenedor junto con sus dependencias y ejecutarlos sobre cualquier equipo que tenga instalada la plataforma Docker, usualmente servidores Linux, nubes públicas y privadas [15]. Docker amplía LXC con un kernel y una Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés) a nivel de aplicación que permiten la ejecución de procesos aislando: CPU, memoria, E/S, red, etc. Además, Docker utiliza diferentes espacios de nombre para aislar por completo la vista de una aplicación del entorno subyacente (Sistema Operativo), lo que incluye los árboles de procesos, la red, los IDs de usuario y los sistemas de archivos [5].

En Docker se distinguen tres componentes principales:

- Imágenes Docker: similares a plantillas de solo lectura mediante las cuales se crean los contenedores. Se les pueden agregar más paquetes y bibliotecas, las cuales son agregadas como capas. Docker permite crear nuevas imágenes así como usar imágenes existentes.

- Registro Docker: es un sistema de almacenamiento y entrega de contenido en el que se encuentran las imágenes Docker. Puede ser público o privado y permite tanto la carga como la descarga de imágenes Docker.

- Contenedores Docker: es una unidad estándar de software que permite encapsular una aplicación y todas sus dependencias para que ésta sea ejecutada. Son creados a partir de imágenes Docker y pueden ser ejecutados, detenidos, movidos y eliminados.

2.1.3 Estilos arquitectónicos de software

Un estilo arquitectónico (también conocido como patrón de diseño) es una estructura fundamental de organización usada en el desarrollo de soluciones de software. Estos patrones se representan mediante un esquema genérico donde se definen y describen los componentes que lo constituyen, sus responsabilidades, relaciones y las formas en que colaboran [42, 50]. Por simplicidad y para evitar confusiones entre un patrón arquitectónico y un estilo arquitectónico [2], en este trabajo se mencionan simplemente como patrones, pero haciendo referencia a estilos arquitectónicos. A continuación se mencionan algunos ejemplos de patrones:

- Filtros y conectores: este patrón permite concatenar múltiples componentes (filtros) mediante un conjunto de interfaces de E/S (conectores). Los filtros son usualmente tratados como cajas negras volviéndolos autónomos e intercambiables, lo cual permite que sean organizados de manera secuencial. Los filtros están basados en el modelo de procesamiento ETL (extracción, transformación y carga [53]) por lo que cuando termina de ejecutarse un filtro, éste puede invocar la ejecución del siguiente o terminar la ejecución del patrón si es el último.

- Manejador/Trabajador: en este patrón existen dos tipos de entidades, mediante las cuales se logra hacer paralelismo de tareas. La entidad manejador es la encargada de dividir una actividad en tareas, las cuales serán encargadas a los trabajadores de la manera que éste decida. Por su parte, un trabajador es el encargado de realizar el procesamiento de las tareas que le sean asignadas por el manejador, siguiendo un modelo de procesamiento ETL. Cada trabajador puede entregar el resultante a una siguiente entidad o a un almacén.

2.1.4 Microservicios

La arquitectura de microservicios se origina como una manera de solventar las deficiencias de las aplicaciones monolíticas permitiendo construir pequeñas aplicaciones o servicios en lugar de una

única aplicación grande. Un microservicio es un proceso (o servicio) cohesivo e independiente que se comunica mediante paso de mensajes a través de un protocolo definido [16]. Si bien la arquitectura de microservicios esta basada en SOA (del inglés, Service-Oriented Architecture) [36], tiene algunas características que establecen diferencias, por ejemplo:

- Tamaño: es comparativamente pequeño. Acorde a la arquitectura, un microservicio no puede ser muy grande, ya que es dependiente de la estructura de diseño. Por lo tanto, debe mantener la granularidad y proveer una tarea pequeña.
- Contexto acotado: se refiere a que las funciones que sean combinadas como un servicio deben estar estrechamente relacionadas.
- Independencia: cada servicio en la arquitectura de microservicios es independiente en la manera operacional de los otros servicios, solo se debe poder comunicar mediante las interfaces definidas.

Las características representativas de los microservicios son las siguientes:

- Flexibilidad: es la capacidad de un sistema para soportar los cambios que le sean realizados a fin de mantenerse actualizado.
- Modularidad: el sistema se conforma de componentes aislados que trabajan en conjunto y cada uno aporta al objetivo, en lugar de tener un solo componente que provee toda la funcionalidad.
- Evolución: el sistema se encuentra constantemente agregando nuevas funcionalidades.

2.1.5 Entrega continua de información

Entrega continua es un concepto utilizado en el desarrollo de software para referirse al conjunto de prácticas que permiten realizar liberaciones (entregas) de los productos de software de manera continuada [7, 8]. Estos productos incluyen valiosas versiones de software en periodos cortos de

tiempo. Para propósitos de este trabajo, por entrega continua nos referimos a garantizar la eficacia de un patrón con respecto a la información provista por el mismo. Esto es que sea capaz producir el efecto deseado en el patrón. Tiene relación con el flujo de datos y de trabajo a través de los componentes de un patrón.

- Flujos de datos: se refiere a la programación de flujo de datos (DFP, del inglés Dataflow Programing). Es un paradigma en el que su ejecución representa el flujo de datos entre los nodos, que puede ser interpretado mediante un grafo dirigido. Este modelo permite ejecutar los datos en cuanto llegan al nodo sin tener tiempos muertos [51].

- Flujos de trabajo: es la automatización de procedimientos en donde la información o tareas son pasados entre los participantes acorde a una serie de reglas para conducir a un propósito especifico o a resolver un problema [44]. Tiene dos enfoques principales: orquestación y coreografía. La orquestación describe cómo deben interactuar los servicios definiendo el control y flujo de datos. La coreografía se realiza de manera colaborativa, describiendo de manera global la organización a seguir en un grupo de servicios para lograr un objetivo común [4].

2.2 Supervisión

La supervisión tiene una estructura jerárquica con al menos dos niveles, siendo el nivel más bajo el encargado del monitoreo. Puede involucrar dos actividades principales: decisión y reconfiguración. En la decisión se provee una opción de resolución o mejora para un caso de falla. La reconfiguración es el proceso de acción en el que se realiza un cambio de estado en los recursos ya sea recolocando o realizando una recuperación [10]. A continuación se mencionan tanto el monitoreo como el almacenamiento que se le da a los datos obtenidos.

2.2.1 Monitoreo

El monitoreo es la recolección de datos de un proceso o componente, esta información sirve para determinar el estado de un sistema. Se limita a solo el manejo de datos, sin abarcar los modelos o patrones involucrados. Tiene como principal actividad la adquisición de datos, aunque también la detección de fallas [10]. En relación con este trabajo de investigación, se distinguen dos enfoques:

- Monitoreo de recursos en computadores físicos: en este enfoque se realiza la recolección periódica de datos tomando como objeto de monitoreo a los computadores físicos. Usualmente se consideran recursos como: CPU, memoria, disco y red. Es importante debido a que es la infraestructura física sobre la que se ejecutan los contenedores virtuales, por lo que pueden estar relacionados directamente.

- Monitoreo de recursos en contenedores virtuales: es otro enfoque de monitoreo que considera las capacidades actuales de virtualización a nivel de software, por lo que su objeto son los contenedores virtuales. Usualmente una plataforma de contenedores permite de forma nativa realizar un monitoreo de sus contenedores [14], aunque también existen diversas alternativas [11, 18, 52].

2.2.2 Indexado de métricas de control

El indexado es el proceso mediante el cual se almacena y se organiza la información de interés, con el objetivo de que las búsquedas posteriores no tomen mucho tiempo en realizarse. El índice invertido es un enfoque tradicional para este tipo de actividad cuando se manipula sólo texto, pero éste involucra más procesos, como tokenización y steaming, permitiendo guardar solo tokens y su ubicación reduciendo el tamaño del índice. Para las métricas utilizadas en la supervisión de contenedores se requiere de una estructura que permita realizar operaciones que resuman dichos datos para cada componente y cada solución monitorizada, por lo que se ha optado por un cubo de

datos.

- **Cubo de datos**: está compuesto de una red o esqueleto de cuboides, donde cada cuboide representa un agregado de atributos (valores) que éste contenga, usualmente mediante la operación *group-by*. Esta operación usa un subconjunto de datos para realizar agregados multidimensionales de tal forma que al pre-calcular dichas operaciones se agilizan las consultas [29]. Un cubo de datos puede representar un almacén de datos como lo es un conjunto multidimensional de bases de datos. Es común usar OLAP (del inglés, On-Line Analytical Processing) en el análisis de los datos en este tipo de estructura [26]. En OLAP se distinguen cuatro operaciones analíticas:

  - *Roll-up*: es una consolidación o agregación que puede ser abordada de dos maneras: mediante reducción de dimensiones o subiendo niveles en la jerarquía (agrupando por algún tipo de orden). Esto es, pasar de datos detallados a datos reducidos.
  - *Drill-down*: es un desglose (operación inversa a Roll-up). Esto puede ser mediante el incremento de dimensiones o bajando niveles en la jerarquía (desagrupando algún tipo de orden).
  - *Slice* y *Dice*: *Slice*, permite seleccionar los datos correspondientes a una dimensión del cubo, como resultante se produce un subcubo. *Dice*, también produce un subcubo pero al seleccionar una o más dimensiones.
  - *Pivot*: realiza la rotación de los ejes de datos, permitiendo diferentes vistas al presentar los datos.

2.3 Representación de soluciones de software

La forma de representar soluciones de software es un tema de especial interés debido a la complejidad inherente que estos conllevan. Esta complejidad se ve incrementada por las nuevas

tendencias de desarrollo de software, como son el uso de patrones y contenedores virtuales. El modelado sirve como conducto para lograr la representación de estas soluciones. La representación se refiere a generar un modelo de las soluciones de software considerando elementos de interés, por ejemplo el comportamiento de los contenedores.

2.3.1 Modelado

El modelado es la acción de generar un modelo mediante la abstracción de un objeto, sistema o fenómeno del mundo real. Esta abstracción se refiere a una simplificación, ya que sólo incluye los detalles relevantes para un propósito determinado. De manera general, un modelo ayuda a la comprensión y comunicación de ideas [34]. En desarrollo de software, un modelo también sirve como ayuda para la implementación del sistema modelado.

Hablar de modelado es hablar de la interpretación del conocimiento, mejor conocido como modelado conceptual, el cual es la base para generar modelos. Éste consiste en la creación, uso y transformación de representaciones mentales. Las representaciones también proveen de una intención, la intención que se desea transmitir [49]. Es llamado conceptual debido a que usa elementos conceptuales (como pensamientos) y no conceptuales (como sensaciones) recibiendo el nombre de conceptualizaciones [43].

Diagrama de Flujo de Datos: DFD, del inglés Data Flow Diagram. Es un método estructurado de diseño y análisis que permite representar visualmente modelos lógicos y la transformación de datos. DFD incluye las siguientes características [33]: 1) sirve de apoyo a la etapa de análisis y requerimientos en el diseño del sistema; 2) es una técnica de representación gráfica con anotaciones; 3) describe una red de actividades / procesos del sistema objetivo; 4) permite comportamientos paralelos y asincrónicos; 5) soporta refinado gradual a través de la descomposición jerárquica de procesos. En DFD, la descomposición de procesos va de arriba hacia abajo desde un nivel general hasta el nivel de detalle deseado. Esto incluye el diagrama de contexto, un diagrama nivel-0 y diagramas secundarios (tantos como sean necesarios). El diagrama de contexto muestra el alcance del sistema, sus límites,

así como las entidades externas que interactúan con el sistema. Por otra parte, el diagrama nivel-0, muestra los procesos principales, el flujo de datos, así como los almacenamientos. Finalmente, los diagramas secundarios (*nivel* > 0), en cada caso van mostrando grados de descomposición. Un DFD usualmente va junto con un diccionario de datos que incluye lo siguiente:

- Nombre: el título del elemento de datos o de control, almacén de datos o entidad externa.
- Alias: nombre alternativo.
- Uso: dónde y cómo usar.
- Representación de contenido: sistema de símbolos para la representación de contenido.
- Información adicional: tipo de datos, valor predeterminado, restricción, etc.

WoT: Del ingles, Web of Things. Describe una serie de estándares que ofrecen la posibilidad de conectar dispositivos del IoT (del inglés, Internet of Things) de una manera personalizable, flexible y eficiente. En lugar de asignar IPs a los dispositivos, permite que hablen el mismo lenguaje, tanto para comunicarse como para interoperar en la Web [56]. Para lograr esta comunicación usa un **modelado** mediante bloques de construcción llamados *Thing Description* (TD). Un TD contiene los metadatos e interfaces de *Things*, donde un elemento *Thing* es una abstracción de una entidad física o virtual que interactúa en el WoT. Si una entidad quiere participar con otras entidades debe proporcionarles su TD. Una entidad receptora debe ser capaz de analizar y comprender los TD a fin de usar la información contenida. Una instancia TD tiene cuatro elementos principales: metadatos textuales, un conjunto de capacidades de interacción o usos, esquemas de intercambio de datos y enlaces Web (expresando la relación con otros *Things* o documentos). Toda esta información dentro del TD está codificada en formato JSON, con la posibilidad de extender sus capacidades mediante JSON-LD [55].

2.3.2 Multi-modelado

El enfoque de Multi-modelado o Modelado Múltiple tiene como objetivo principal la representación de sistemas físicos para lograrlo se vale de utilizar diversos modelos del sistema deseado. Estos modelos están basados en diferentes conocimientos como son conceptos ontológicos, premisas de representación, tipos epistemológicos y niveles de agregación, los cuales son agrupados de la siguiente manera (Figura 2.1):

1. Conceptos ontológicos: a nivel de componentes (independientes del contexto/sistema) o a nivel de subsistemas (unidades organizadas que no pueden definirse de manera aislada).

2. Premisas de representación: las cuales limitan el alcance y la precisión del modelo a obtener.

3. Tipos de conocimiento:

    - Estructural: referente a la topología del sistema.
    - Comportamiento: de cómo trabajan los componentes de acuerdo a variables y parámetros.
    - Funcional: el rol que el componente tiene en el sistema.
    - Teleológico: el objetivo o meta a lograr de acuerdo a las condiciones de operación.

4. Niveles de agregación: se refiere a la granularidad deseada, esto es si se representará a nivel de subsistema o componente.

El enfoque de multi-modelado permite elegir qué tipo de ontología, tipo de representación, tipo epistemológico y/o nivel de agregación usar. Sin embargo, originalmente trata las técnicas para la representación de conocimiento estructural y de comportamiento, enfocándose en la función y teleología. Esto se debe a que con el conocimiento teleológico se abarca la parte del propósito para el que fue hecho el sistema y con el conocimiento funcional se pueden relacionar esos propósitos con la estructura del sistema a través de sus procesos y roles funcionales. La idea de poder utilizar

los diversos modelos consiste en aplicarlos en la resolución de problemas como son la interpretación, diagnostico, simulación, entre otros.

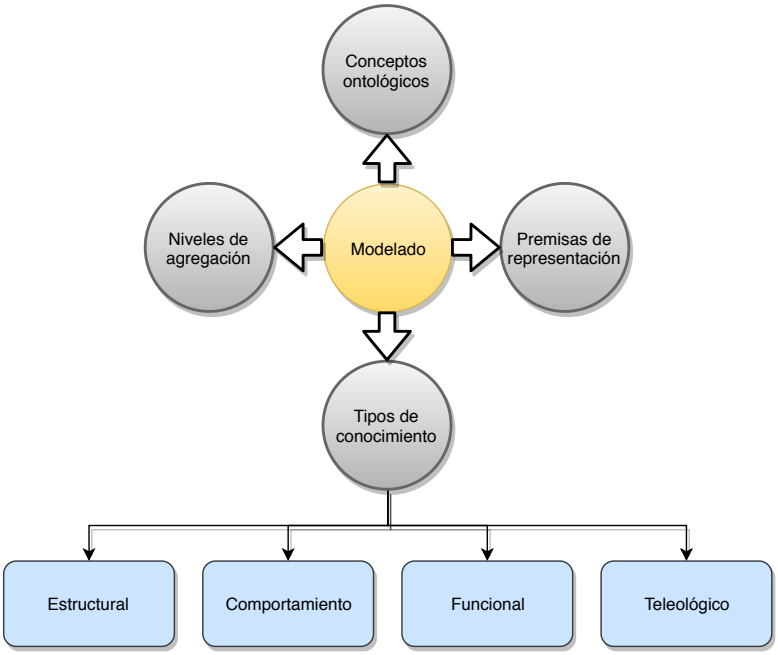


Figura 2.1: Taxonomía de modelado del conocimiento conforme a la propuesta de multi-modelado [9].

3

Estado del arte

En este capítulo se presenta una recopilación y análisis de los principales trabajos en la literatura relacionados al tema de investigación abordado en esta tesis. La Sección 3.1 menciona algunos trabajos de monitoreo enfocados en la nube y que adicionalmente utilizan tecnologías de virtualización. La Sección 3.2 describe propuestas que involucran el uso de estilos arquitectónicos de software, así como la entrega continua de información. En la Sección 3.3 se aborda la representación de sistemas basados en contenedores virtuales. Finalmente, en la Sección 3.4 se realiza un resumen comparativo de las diferentes propuestas presentadas en este capítulo.

3.1 Mecanismos de monitoreo

El monitoreo es un proceso que juega un rol importante en el seguimiento y vigilancia de infraestructuras de cómputo, se encarga de recolectar información referente al uso de sus recursos, esto mientras se ejecutan sistemas o aplicaciones que resuelven problemas específicos. Posteriormente la información recolectada suele ser usada para la toma de decisiones informadas. El monitoreo se

puede categorizar de diversas formas [21]: de propósito general, enfocado en la nube y Gestión del Rendimiento de Aplicaciones (APM, por sus siglas en inglés) [27]. Pero independientemente de la categoría, éste involucra las actividades de recolección, almacenamiento, análisis y presentación de la información.

Matthew et al. [37] propusieron Ganglia, un sistema de monitoreo robusto para sistemas distribuidos de alto rendimiento como clústeres y *grids*. Ganglia se basa en un diseño jerárquico dirigido a las federaciones de clústeres. Este diseño utiliza un árbol para representar los nodos y su estado. En la arquitectura de tipo árbol que presenta Ganglia, una hoja representa un nodo, mientras que los nodos ubicados en la parte superior especifican puntos de agregación. Como cada nodo mantiene la información de monitoreo, cada hoja representa lógicamente un clúster distinto y cada nodo no hoja representa un conjunto de clústeres. Este enfoque se basa en un protocolo de multidifusión de aviso/escucha para monitorear punto a punto el estado de los nodos. Además, aprovecha tecnologías ampliamente utilizadas como XML[1] para la representación, XDR[2] para la portabilidad de datos compactos y RRDTool[3] para el almacenamiento y visualización de datos. Para el monitoreo utiliza una serie de mensajes sobre una dirección de multidifusión que se puede interpretar como un protocolo de membresía, la cual es mantenida al hacer la recepción de esos mensajes. Se considera como activo al nodo que recibe los mensajes y como inactivo al nodo que no los recibe después de varios intentos en un periodo corto de tiempo. Cada nodo recibe y envía paquetes que contienen información de monitoreo, esta información también puede contener métricas específicas de la aplicación. De esta manera, todos los nodos tienen una vista aproximada del estado en el que ese encuentra el clúster completo. Si bien Ganglia realiza un monitoreo de alta calidad, éste tiene un enfoque en equipos físicos. En el trabajo de esta tesis no nos enfocamos en los equipos físicos, sin embargo, la forma en que está desarrollado Ganglia es de nuestro interés. Por ejemplo, el que utiliza dos tipos de demonios para realizar su funcionamiento: uno como línea de comandos por cada

[1]Extensible Markup Language
[2]External Data Representation
[3]Round Robin Database Tool

clúster y otro en cada nodo del clúster encargado de proveer las métricas.

Bertolino et al. [6] presentaron GLIMPSE, una infraestructura de monitoreo flexible basada en eventos. GLIMPSE implementa los complementos clave para realizar la detección de eventos mediante un patrón de mensajes de publicación/subscripción. Para abordar la flexibilidad lo hace a alto nivel desacoplando cómo se especifican los eventos a la forma en como éstos se observan y analizan. El desacomplamiento se puede observar en la arquitectura, la cual está compuesta por los elementos que se describen a continuación:

- Pruebas: éstas interceptan los eventos que ocurren en el software y los envían al bus de monitoreo.
- Bus de monitoreo: es el medio de comunicación por el que fluye toda la información (eventos, preguntas, respuestas).
- Procesador de eventos: un motor de reglas que analiza los eventos para inferir otros nuevos que encajen con la solicitud del usuario.
- Consumidor: que puede ser un motor de aprendizaje, un analizador de dependencias o un simple usuario que solicita información.
- Gestor: es el encargado de orquestar toda la arquitectura de GLIMPSE.

GLIMPSE fue desarrollada con base en una arquitectura genérica de monitoreo bien definida, en la que se especifican y describen las funcionalidades que se pueden incluir, como son: recolección, interpretación, transmisión y presentación de los datos. Estas funcionalidades se relacionan con la propuesta de este trabajo de investigación, la cual además incluye una representación de las aplicaciones monitorizadas.

Dynatrace [19] es un proyecto que realiza Gestión del Rendimiento de Aplicaciones (APM), monitoreo en la nube, entre otros servicios. Proporciona herramientas para el monitoreo del rendimiento de aplicaciones basadas en la Web, así como para infraestructura en la nube, por lo que

abarca tecnologías como contenedores y máquinas virtuales. Para realizar la recolección de métricas utiliza un programa llamado *OneAgent*, el cual requiere tenerlo instalado en cada máquina que sea parte de la infraestructura. Mediante la tecnología *Smartscape* detecta los diferentes artefactos de una aplicación, lo que permite proporcionar una representación gráfica de la topología que existe entre ellos. Sin embargo, esta topología se realiza a nivel de contenedores y no contempla el nivel de solución de software o conjunto de contenedores. Por lo que éste enfoque no contempla el patrón que implementa una solución ni la importancia que tenga en la misma.

Prometheus [22] es un proyecto de código abierto mantenido por *Cloud Native Computing Foundation* desde 2016. Se enfoca en el monitoreo y alerta de infraestructura física y de contenedores virtuales. Esto lo realiza mediante componentes especializados para realizar tareas que van desde la extracción de métricas (o *scraping*), almacenamiento de las mismas mediante un modelo multidimensional de datos de series de tiempo y finalizando con la visualización de datos, así como con el lanzamiento de alertas. Cada actividad es realizada mediante un componente acoplable que puede ser intercambiado o combinado con otros componentes compatibles. Por ejemplo, para el *scraping* usa *Pushgateway* que sirve como puente, para obtener métricas mediante la técnica de *push*. Mediante la técnica *pull* puede obtener las métricas de componentes externos como *cAdvisor*[4] para contenedores virtuales y NodeExporter para el equipo físico. La visualización de datos es realizada mediante *Grafana* y las alertas se manejan mediante el componente *Alertmanager*. Éste permite la discretización de alertas con técnicas de de duplicidad, agrupado y ruteo con otras integraciones como *email*, *PagerDuty*, entre otros. La principal diferencia con la propuesta de este trabajo de investigación es que Prometheus desacopla sus componentes permitiendo una mejor integración con otros componentes. Sin embargo esta generalización delega actividades mientras se enfoca en la obtención, almacenamiento y presentación de datos de rendimiento en puro; es responsabilidad del usuario interpretar o procesar estos datos.

AppFormix [40] es un proyecto de *Juniper Networks* que permite monitoreo, calendarización y

[4]https://github.com/google/cadvisor

gestión del rendimiento para infraestructura definida por software. Esto incluye monitoreo histórico y de tiempo real, así como gráficos de rendimiento. Puede operar en entornos de nube privada, pública e híbrida como son *OpenStack* [41], *Azure* [38] y *Amazon Web Services* (AWS[5], por sus siglas en inglés), así mismo considerando a los contenedores virtuales y máquinas virtuales. En este proyecto se aprovecha la naturaleza del cómputo distribuido para analizar y detectar cuando un SLA (del inglés *Service Level Agreement*) no se cumple, tanto para aplicaciones como para carga de trabajo. Respecto al funcionamiento, utiliza un *Smart Agent* que se ejecuta sobre cada nodo de la infraestructura de nube. Es mediante este agente que se realizan las mediciones para extraer las métricas. Además, utiliza los recursos locales al ejecutar algoritmos de aprendizaje máquina para detectar anomalías y datos atípicos. Esto permite discretizar la cantidad de información que se transmite a un punto central de recolección, reduciendo la cantidad de información almacenada y aligerando el proceso de análisis. Si bien considera tecnologías de virtualización, como son los contenedores virtuales, sigue teniendo un enfoque general a nivel de infraestructura física solo considerando los contenedores como unidades independientes y no que puedan estar relacionados en un patrón.

3.2 Estilos arquitectónicos y entrega continua de información

Un estilo arquitectónico consiste en seguir un diseño o esquema definido en el desarrollo de software. Mientras que la entrega continua permite realizar y entregar mejoras de un servicio de forma rápida, eficiente y fiable. Referente a la información, la idea básica es tomar las características de la entrega continua y aplicarlas a la entrega continua de información. Estos conceptos cada vez son más utilizados en el desarrollo de software. A continuación se presentan propuestas que utilizan o permiten capturar y mostrar alguno de estos conceptos.

Yañes et al. [24] propusieron Sacbe, una arquitectura de software modular que permite encapsular

[5]https://aws.amazon.com/

componentes de software en Building Blocks (BB, por sus siglas en Inglés), los cuales son una representación lógica de aplicaciones independientes encapsuladas en contenedores. En Sacbe estos BB han sido diseñados para producir un patrón de software arquitectónico (encadenando los BB a través de interfaces E/S) que incluye las etapas de adquisición, procesamiento y entrega. En la etapa de adquisición, un BB extrae los datos de una fuente de datos u otra aplicación. En la etapa de procesamiento los datos son procesados por instancias de software para producir una versión asegurada de los mismos. Finalmente, en la fase de entrega, la versión asegurada de los datos es enviada a una ubicación destino. El encadenamiento de múltiples BB produce una ruta virtual que se interpreta como un flujo continuo de los datos/metadatos desde una fuente hasta un destino. En este tipo de software modular se facilita el intercambio de componentes gracias a los BB, pero surge el problema del incremento de complejidad para seguir la ruta de los datos y principalmente de identificar un punto de falla entre los bloques.

Armenise et al. [1] propusieron Jenkins, el cual se ha convertido en un orquestador para los equipos de desarrollo involucrados en el ciclo de vida del software. Inició como una herramienta de integración continua de código abierto, lo que implicaba realizar las diferentes etapas de un flujo de trabajo para crear, probar e implementar componentes de software. Posteriormente, cambió su enfoque al de una plataforma de entrega continua, permitiendo también automatizar los pasos realizados por la línea de equipos. Estos pasos van desde el chequeo del código, pasando por las pruebas unitarias / de rendimiento a la liberación de los binarios, hasta el despliegue en producción. Esto se logra gracias a sus complementos que hacen posible encadenar las tareas, promover su ejecución y permitir la intervención humana. Es importante realizar la trazabilidad de estas etapas a fin de validar su correcta ejecución, lo cual ayuda a detectar la causa de los errores en una ejecución fallida. En Jenkins una manera de hacerlo es mediante sus complementos de visualización que muestran el resultado de la ejecución de un *pipeline*. Esta forma de realizar la detección de fallas se interpreta de manera semejante a detectar fallas sobre una solución de software implementada siguiendo un patrón. Esto debido a la similitud con el pipeline antes mencionado. Una desventaja de la trazabilidad que realiza

Jenkins radica en el tiempo de ejecución, ya que éste es realizado al momento de compilación y tiene la limitante de ser una captura de ese instante.

3.3 Representación de sistemas basados en contenedores virtuales

Una representación de modelado consiste en realizar un modelo abstracto de un fenómeno o sistema. En este caso se trata de ambos, el sistema es una solución de software implementada mediante contenedores virtuales y un patrón. Mientras que el fenómeno es la detección del estado de dicha solución. A continuación, se muestra una propuesta que cumple con las características buscadas en este trabajo de investigación.

Chittaro et al. [9] presentaron una propuesta de multi-modelado para la representación de sistemas físicos. Para las tareas de razonamiento toma en cuenta varios modelos en el que cada uno se enfoca en un tipo específico de conocimiento, lo que permite tomar ventaja de cada modelo generado. Los tipos de modelos son: estructural, de comportamiento, funcional y teleológico. Estos modelos se pueden organizar de la siguiente manera:

- Conocimiento fundamental:
  - Estructural: se refiere al conocimiento de la topología de un sistema y de sus componentes.
  - Comportamiento: es el conocimiento de cómo trabajan los componentes de acuerdo a las variables y los parámetros que caracterizan un estado.
- Conocimiento interpretativo:
  - Funcional: es el conocimiento de los roles que un componente puede tener dentro de un sistema. Este tipo de conocimiento relaciona el comportamiento del sistema con sus objetivos.

- Teleológico: conocimiento respecto al propósito con el que fue creado un sistema, así como de sus condiciones operacionales.

Esta propuesta de multi-modelado ha sido utilizado en diversas tareas como: interpretación, diagnóstico, diseño entre otras, por ello y debido a sus características, consideramos que es útil en las tareas de modelado en entornos virtuales. En este trabajo se utiliza una adaptación de dicho modelado, lo que corresponde a la interpretación de la estructura, comportamiento y funcionalidad como parte del diagnóstico de anormalidades para soluciones basadas en contenedores virtuales.

El FP7-ICT [23] desarrollaron *Internet of Things Architecture* (IoT-A) *Architecture Reference Model* (ARM) o IoT-A ARM [45] una propuesta para el modelado de entornos IoT como un solo sistema. Está compuesta de dos principios: modelo de referencia y arquitectura de referencia. El modelo de referencia sirve para establecer un entendimiento del dominio del IoT mediante modelos que identifican conceptos importantes, interacciones y restricciones. Mientras que la arquitectura de referencia son una serie de guías, buenas prácticas y vistas que pueden ser usadas para materializar arquitecturas concretas y sistemas IoT operables.

La arquitectura de un sistema es una herramienta de comunicación entre las diferentes partes. Por ejemplo, los desarrolladores, socios y clientes tienen diferentes requerimientos del mismo sistema. Esto resulta en una arquitectura con múltiples representaciones. La arquitectura de un nivel superior es llamada arquitectura de referencia y es la que sirve de base para materializar sistemas IoT. Esto debido a la intención respecto al IoT, la cual es mapear objetos o lugares del mundo real a una representación digital, misma con la que posteriormente se puede interactuar mediante software.

Esta propuesta implica el modelado funcional en cada uno de sus dos principios (modelo funcional y vista funcional). En el modelo funcional (del modelo de referencia) para describir los diferentes grupos funcionales con su relación al ARM y la vista funcional (de la arquitectura de referencia) para describir los componentes funcionales de un grupo funcional, así como las interfaces y relaciones entre componentes. La relación que tiene este modelado respecto a la propuesta del trabajo de

investigación es que modela una entidad pasándola del mundo real al mundo virtual, volviéndolo utilizable según sus intereses. Sin embargo, este modelado es con el propósito principal de generar entendimiento entre las partes involucradas mientras que en nuestro caso es todavía para abstraer el estado de los componentes de una solución de software.

3.4 Resumen del estado del arte

En la Tabla 3.1 se presenta un resumen de las propuestas antes mencionadas. Las propuestas marcadas con los números uno al cinco corresponde a la Sección 3.1, la seis y siete a la Sección 3.2 y la ocho a la Sección 3.3. De los mecanismos de monitoreo, tres de las ocho propuestas tienen como enfoque la nube y los contenedores virtuales, siendo estas: Dynatrace, Prometheus y AppFormix. Además, otras dos propuestas se enfocan en infraestructura física, éstas son: Ganglia y Glimpse. Por la parte de los patrones y entrega continua de datos está Jenkins y Sacbe enfocados en Flujos de trabajo, donde Sacbe también extiende a la nube. Finalmente, se encuentran las propuestas de Chittaro et al. y IoT-A ARM para la representación de sistemas físicos y entornos IoT.

Para las propuestas antes mencionadas y la nuestra se están tomando los siguientes cinco puntos a considerar:

1. Confiabilidad: es capaz de identificar el atributo de Confiabilidad.
2. Eficiencia: es capaz de identificar y/o mostrar el atributo de Eficiencia.
3. Patrones: opera a nivel de patrones o es capaz de identificarlos. Donde un patrón abarca un conjunto de contenedores relacionados entre sí.
4. Contenedores virtuales: opera a nivel de contenedor virtual o es capaz de identificarlo.
5. Modelado de sistemas: realiza técnicas de modelado sobre sistemas contemplando sus componentes.

#	Referencia	Descripción	Enfoque	Supervisión de		Orientado a		
				(1)	(2)	(3)	(4)	(5)
1	Ganglia [37]	Sistema de monitoreo	Clúster y Grid	✓	✓	×	×	×
2	Glimpse [6]	Infraestructura de monitoreo	Sistemas heterogéneos	✓	✓	×	×	×
3	Dynatrace [19]	APM[6] y monitoreo	La nube, CVs, MVs	✓	✓	×	✓	×
4	Prometheus [22]	Monitoreo y TSDB[7]	La nube, CVs, MVs	✓	✓	×	✓	×
5	AppFormix [40]	Herramienta de monitoreo	La nube, CVs, MVs	✓	✓	×	✓	×
6	Jenkins [1]	Orquestador de entrega continua	Flujos de trabajo	×	×	×	✓	×
7	Sacbe [24]	Almacenamiento de datos	Flujos de trabajo en la nube	×	×	✓	✓	×
8	Chittaro et al. [9]	Representación	Sistemas Físicos	×	×	×	×	✓
9	IoT-A ARM [45]	Representación	Entornos IoT	×	×	×	×	✓
10	Propuesta	Supervisión	SECI[8]	✓	✓	✓	✓	✓

Tabla 3.1: Resumen de propuestas en el estado del arte.

Se ha observado que las propuestas bien posicionadas de monitoreo realizan de forma acertada la presentación de métricas de estado. Esto para equipos físicos y tecnologías de virtualización como son los contenedores y las máquinas virtuales. En algunos casos se realizan agregados antes de presentar dichas métricas e incluso al momento de la recolección para reducir la cantidad de información transmitida y almacenada. Esto aunado a más características de cada propuesta, por ejemplo, el manejo de alertas o la capacidad de acoplar otros componentes. Sin embargo, solo propuestas de propósitos específicos como Jenkins y Chittaro et al. realizan por separado las tareas buscadas en este trabajo de investigación. Estas tareas corresponden a la representación o modelado del estado de una solución, la cual contempla un conjunto de contenedores.

4

Diseño y desarrollo de un método para la supervisión de sistemas de contenedores virtuales basado en multi-modelado

En este capítulo se describe en la sección 4.1 el diseño del método para la supervisión de sistemas de contenedores virtuales incluyendo los módulos monitoreo, indexado, representación y diagnóstico. La sección 4.2 muestra las dos arquitecturas usadas para comprender la relación entre dichos módulos, siendo éstas una global y una específica. Por último, en la sección 4.3 se describe la implementación de los módulos del método de suspervisión.

4.1 Diseño de la propuesta

En esta tesis se propone un método para la supervisión del estado de una solución de software (aplicación o aplicaciones) que se ejecuta sobre contenedores virtuales. La idea principal de realizar la supervisión de este tipo de soluciones es que facilite la interpretación del estado en el que se encuentra una solución en determinado momento. Para ello se requiere enriquecer la información que se presenta al usuario final. Esto implica que la información de una solución sea procesada por los módulos del método propuesto.

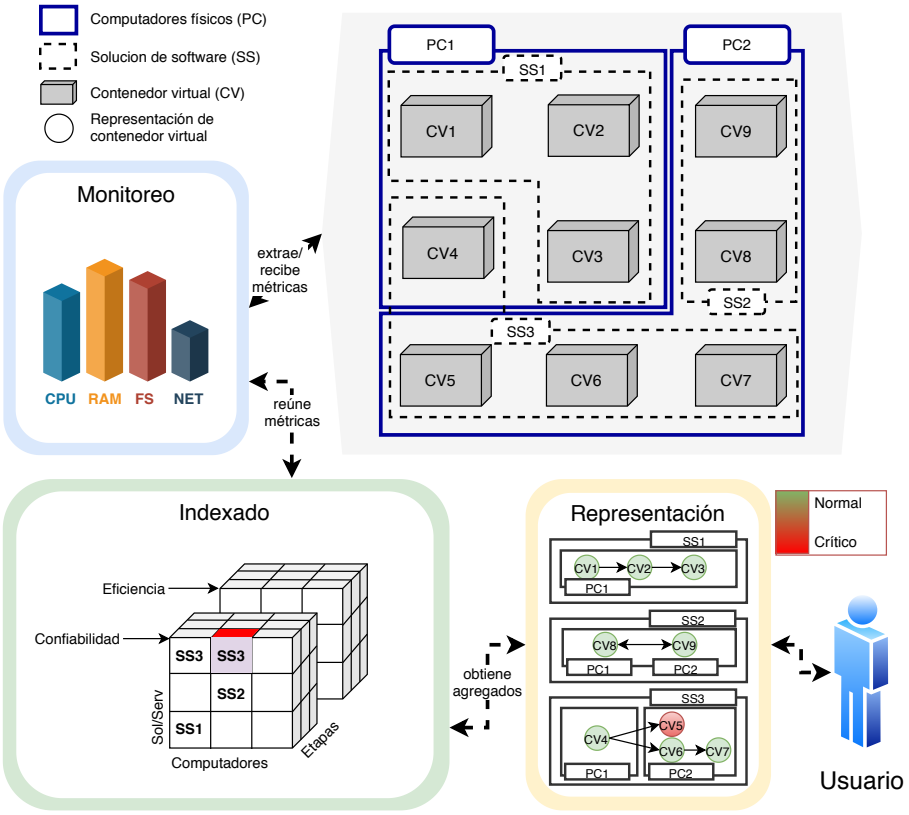


Figura 4.1: Representación conceptual del método propuesto.

En la Figura 4.1 se muestra una representación conceptual del método propuesto, donde se tiene el entorno en el cual viven las soluciones de software y los módulos del método de supervisión. Para conocer el estado de una solución, ésta se debe agregar al sistema de supervisión y pasar por sus

etapas. El sistema de supervisión está compuesto de 4 módulos principales: Monitoreo, Indexado, Representación y Diagnóstico. Primero, se tiene un conjunto de contenedores virtuales, marcados con etiqueta *CV*. Estos contenedores se ejecutan sobre uno o más computadores físicos, marcados con la etiqueta *PC*. A partir de esto se incluyen los módulos descritos a continuación:

- Monitoreo: en este módulo se realiza la extracción / recepción de métricas de cada contenedor que se encuentra en cada computador físico. Además, se definen las métricas de interés, la frecuencia de obtención y formato. Posteriormente, dichas métricas son enviadas al módulo de Indexado. La definición de la frecuencia de obtención de los datos se hace de acuerdo a la evaluación experimental como se muestra en el Capítulo 5.

- Indexado: en este módulo se efectúa el almacenamiento de la información, tanto de las soluciones, los contenedores y las métricas, así como del estado de una solución en determinado momento. De forma que esta información se encuentra disponible para ser consultada por otros módulos. Por ejemplo, por el servicio de Representación para presentarse a un usuario final.

- Representación: en este módulo se genera una representación abstracta por solución, la cual incluye el modelado de los componentes y la relación con su estado actual. Este módulo es alimentado con la información de una solución, la cual es obtenida mediante una consulta al servicio de Indexado.

- Diagnóstico: en este módulo se obtiene el estado en el que se encuentra una solución en un período de tiempo determinado. El período es un rango que va desde un tiempo de inicio, hasta un tiempo final. Para obtener el estado primero se definen umbrales y escalas. Donde los umbrales marcan límites sobre los cuales se presenta un comportamiento y las escalas son la magnitud que representa dicho umbral. En este caso los umbrales utilizados son los rangos 0-0.33, 0.34-0.66 y 0.67-1. Donde 0 es una utilización nula y 1 la máxima utilización. Se consideran tres escalas: bajo, medio y alto, por cada rango respectivamente. Estos rangos se mencionan nuevamente en la sección 4.1.4.

4.1.1 Módulo de monitoreo

El módulo de Monitoreo se encuentra en interacción directa con las soluciones y sus componentes. Éste es el encargado de la extracción / recepción de las métricas de cada componente. Primero, hay que comprender la jerarquía que existe entre las entidades involucradas en esta actividad.

En la Figura 4.2 se muestra la jerarquía de entidades en el modelo de Monitoreo. Se tiene una entidad **H** que corresponde a los computadores físicos (Ecuación 4.1), una entidad **R** (Ecuación 4.2) representando a los recursos y una entidad **M** (Ecuación 4.3) para las métricas. Estas entidades tienen la relación jerárquica que se muestra en las siguientes ecuaciones. En la Ecuación 4.4 se representa que cada computador físico tiene un subconjunto de los recursos. En la Ecuación 4.5 se expresa cómo cada recurso tiene un subconjunto propio de las métricas, esto es, que no puede tener todas las métricas. Por ejemplo, tomando el caso de la Figura 4.2 cada contenedor *VC1*, *VC2* y *VC3* tiene los recursos CPU, RAM, FS y NET. Cada uno de ellos tiene sus *correspondientes* métricas, pero sin mezclarse con las de los otros recursos.

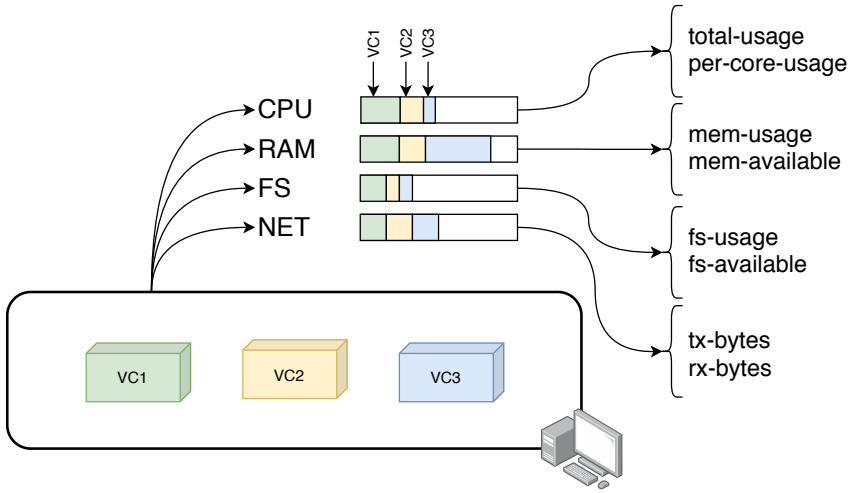


Figura 4.2: Jerarquía del modelo de monitoreo.

- **H**: computadores físicos

$$H = \{r_1, r_2, \ldots, r_j\} \tag{4.1}$$

  - **R**: recursos

$$R = \{CPU, RAM, FS, NET\} \tag{4.2}$$

    - **M**: métricas

$$M = \{total\text{–}usage, per\text{–}core\text{–}usage, ..., m_k\} \tag{4.3}$$

- Relación:

$$\forall h \in H : h = \{r \subseteq R\} \tag{4.4}$$

$$\forall r \in R : r = \{m \subset M\} \tag{4.5}$$

Continuando con las entidades, se tiene **CV** (Ecuación 4.6) para el conjunto de contenedores virtuales y **TA** (Ecuación 4.7) para las transformaciones que se realizan a los datos mediante las aplicaciones que se ejecutan dentro de cada contenedor virtual.

- **CV**: contenedores virtuales

$$CV = \{vc_1, vc_2, \ldots, vc_i\} \tag{4.6}$$

- **TA**: transformaciones (aplicaciones)

$$TR = \{tr_1, tr_2, \ldots, tr_l\} \tag{4.7}$$

Lo anterior encamina a expresar la relación entre contenedores virtuales, computadores físicos y transformaciones. Para ello se presentan los siguientes predicados, que indican que un contenedor es ejecutado sobre un computador físico (Ecuación 4.8), el cual consume una porción de los recursos del mismo (Ecuación 4.9) mientras realiza una o más transformaciones (Ecuación 4.10).

- Predicado $runs\_on$ sobre $CV \times H$.

$$\forall vc \in CV \exists h \in H[runs\_on(vc, h)] \tag{4.8}$$

- Predicado $portion$ sobre $CV \times R$.

$$\forall vc \in CV \exists r' \in R[portion(r) \Leftrightarrow r'] \tag{4.9}$$

- Predicado $apply$ sobre $CV \times TR$.

$$\forall vc \in CV \exists tr \in TR[apply(vc, tr)] \tag{4.10}$$

La obtención de los datos se realiza utilizando una estructura que es compartida con el módulo de indexado. La definición y descripción de esa estructura se realiza en el siguiente módulo.

4.1.2 Módulo de indexado

Una vez entendida la relación entre las entidades, se define una forma de mapear las entidades con los valores de las métricas. Esta forma de mapeo representa una ruta que permite acceder a los valores que requiere obtener el módulo de Monitoreo, mismos que son almacenados en el módulo de Indexado. Una forma de hacerlo se presenta a continuación.

Primero se definen los tipos de recursos, uno para nodos (Ecuación 4.11) y otro para conectores (Ecuación 4.12). Los recursos característicos de un nodo son: *CPU*, *RAM* y *FileSystem*. Mientras

que para un conector los recursos son: *RAM*, *FileSystem* y *Network*. Después, se define el mapa para encontrar cada valor de cada métrica k, de cada recurso j, de igual forma para cada nodo (Ecuación 4.13) y cada conector (Ecuación 4.14). Por último, para obtener los valores por solución se generan mapas de duplas (Ecuación 4.15) por cada contenedor i de la solución y (Ecuación 4.16). De esta forma se puede utilizar este mapeo para obtener los recursos, métricas y valores por cada solución que se agregue al sistema de supervisión.

- Recursos por nodo-conector:

$$RnodeVC_i = \{r_{CPU}, r_{RAM}, r_{FS}\} \tag{4.11}$$

$$RedgeVC_i = \{r_{RAM}, r_{FS}, r_{NET}\} \tag{4.12}$$

- Estado por CV como nodo-conector:

$$STnodeVC_{i(j,k)} = \mathbf{RnodeVC}_1^j \rightarrow \mathbf{m}_1^k \rightarrow value \tag{4.13}$$

$$STedgeVC_{i(j,k)} = \mathbf{RedgeVC}_1^j \rightarrow \mathbf{m}_1^k \rightarrow value \tag{4.14}$$

- Estado por solución:

$$STMapVC_i = \{STnodeVC_{i(j,k)}, STedgeVC_{i(j,k)}\} \tag{4.15}$$

$$STSol_y = \{STMapVC_1, STMapVC_2, \ldots, STMapVC_i\} \tag{4.16}$$

donde:

r = Es un elemento del conjunto de recursos de (Ecuación 4.2).

m_k = Es un elemento del conjunto de métricas de (Ecuación 4.3).

$STSol_y$ = Representa el conjunto de mapas para obtener los valores de cada métrica para nodos y para conectores de una solución y.

Los mapas de estado de una solución son llamados de esta forma ya que los valores que contienen estos mapas son representativos del estado en el tiempo específico en que fueron tomados. Además, estos mapas permiten una única forma de transportar los datos desde el módulo de Monitoreo al módulo de Indexado.

4.1.3 Módulo de representación

En este módulo se realiza el modelado de los componentes de una solución, considerando los enfoques mencionados a continuación. En la Figura 4.3 se muestra un diagrama de los enfoques utilizados para generar la representación, se incluyen: Diagrama de flujo de datos (*DFD*, por sus siglas en Inglés), Multi-modelado y un estándar del Web de las Cosas (*WoT*, por sus acepción en Inglés). Aplicando estos enfoques se obtiene una *Ficha* representativa de la solución objeto. La Ficha contiene los elementos definidos en (Ecuación 4.17). *Contexto* y *Secuencia* corresponden al DFD. Mientras que los elementos *Estructural*, *Comportamiento* y *Funcional* son del enfoque de Multi-modelado. El estándar de WoT aporta las especificaciones que permiten que el modelado sea aceptado tanto para equipos físicos como abstractos, tal es el caso de los contenedores virtuales.

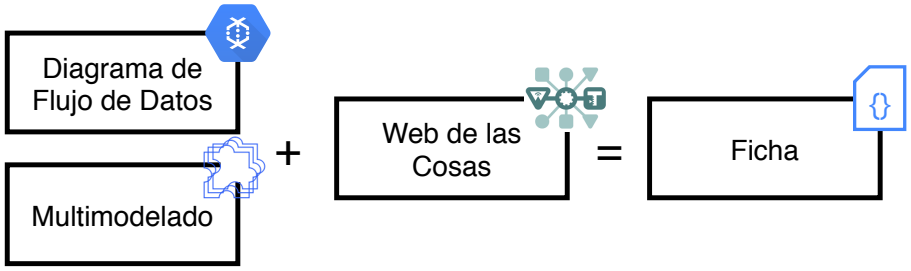


Figura 4.3: Diseño conceptual de la representación generada por el método propuesto.

donde:

$$SystemCard = JSON = \{context, secuential, structure, behaviour, functional\} \quad (4.17)$$

Para ello primero se debe obtener la información de las métricas y de los recursos de una solución.

Esto se realiza mediante una consulta al servicio de indexado, solicitando la cantidad de muestras o un período de tiempo, esto es el tamaño de ventana. En la Ecuación 4.18 se expresa la relación entre la información obtenida del módulo Monitoreo y los enfoques de modelado. Esto es, a partir de los mapas de estado de una solución y y los enfoques de modelado, produciendo una Ficha representativa del estado de una solución.

$$Model(STSol_y, Modeling) = SystemCard \tag{4.18}$$

$$Representation(STSol_y, SystemCard) \tag{4.19}$$

donde:

$STSol_y$ = Son los mapas de estado de una solución y.

$Modeling$ = Son los enfoques de modelado mencionados en la Figura 4.3.

$Model$ = Función que realiza el modelado de la solución y para producir su Ficha representativa.

$SystemCard$ = Es la Ficha resultante mostrada en la Ecuación 4.17.

$Representation$ = Función encargada de materializar el estado de una solución y con su Ficha representativa.

4.1.4 Módulo de diagnóstico

El módulo de Diagnóstico es el encargado de evaluar y obtener el estado en el que se encuentra un sistema incluyendo sus componentes. Esto con base en el factor de utilización que se obtenga en cada recurso evaluado. Los factores de utilización son valores que se encuentran entre 0 y 1. Donde 0 no representa uso del recurso y 1 representa uso completo del mismo. Es necesario establecer los umbrales con los cuales se van a estar evaluando dichos factores y sus correspondientes escalas. Un umbral establece los límites entre los cuales se presenta un comportamiento. Una escala es la magnitud que representa que un comportamiento se encuentre entre determinado rango de valores.

Para este trabajo se tomó como base el ISO para manejo de riesgos *ISO/IEC 73* [35] que establece las escalas de riesgo mostradas en la Figura 4.4, siendo éstas: Bajo (entre 0 y 0.33), Medio (entre 0.33 y 0.66) y Alto (entre 0.66 y 1).

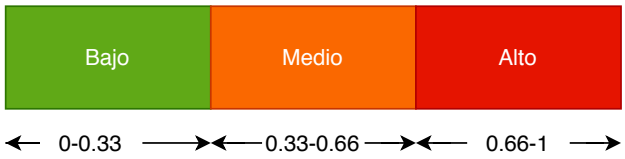


Figura 4.4: Umbrales y escalas de riesgo de tres rangos de acuerdo al *ISO/IEC 73*.

Una vez definidos los umbrales y escalas de riesgo es necesario que los valores de cada recurso sean mapeados a valores entre 0 y 1. Eso se logra al calcular el factor utilización de cada uno de ellos, como se muestra en las siguientes fórmulas. En la Ecuación 4.20 se define el cálculo de la utilización del CPU. En este caso se considera el resumen de todos los núcleos en lugar de considerarlos de forma independiente. En la Ecuación 4.21 se define el cálculo de utilización para el sistema de archivos, donde es común que se cuenten con varias particiones de almacenamiento. Por eso se agrega una sumatoria, desde 1 hasta el número de particiones. En la Ecuación 4.22 se define el cálculo de utilización de la memoria RAM. Por último, en la Ecuación 4.23 se muestra cómo calcular la utilización de la red. Esta métrica se basa en la cantidad de bytes que son tanto recibidos como transmitidos.

El factor de utilización del CPU de un contenedor en un instante de tiempo t, se define con la Ecuación 4.20.

$$U\_CPU = 1 - \left[\frac{T_{CPU} - C_{CPU}}{T_{CPU}}\right] \tag{4.20}$$

donde:

T_{CPU} = Es la capacidad total de procesamiento del computador físico, dada por la sumatoria de la capacidad de cada uno de los núcleos.

C_{CPU} = Es el uso del CPU en un instante actual.

El factor de utilización del sistema de archivos de un contenedor en un instante de tiempo t, se

define con la Ecuación 4.21.

$$U\_FS = 1 - \left[\sum_{i=1}^{f}\left(\frac{T_{FS_i} - C_{FS_i}}{T_{FS_i}}\right)\right] \tag{4.21}$$

donde:

f = Es la cantidad de particiones disponibles en el computador físico.

T_{FS_i} = Es la capacidad total de la partición actual en el computador físico.

C_{FS_i} = Es el consumo de la partición actual en un momento dado.

El factor de utilización de la memoria RAM de un contenedor en un instante de tiempo t, se define con la Ecuación 4.22.

$$U\_MEM = 1 - \left[\frac{T_{MEM} - C_{MEM}}{T_{MEM}}\right] \tag{4.22}$$

donde:

T_{MEM} = Es el total de memoria en el equipo físico.

C_{MEM} = Es el consumo de memoria en el instante t.

El factor de utilización de la red de un contenedor en un instante de tiempo t, se define con la Ecuación 4.23.

$$U\_NET = 1 - \left[\frac{T_{NET} - (TX_{NET} + RX_{NET})}{T_{NET}}\right] \tag{4.23}$$

donde:

T_{NET} = Es la capacidad total de la red en bytes.

TX_{NET} = Es la cantidad de bytes transmitidos.

RX_{NET} = Es la cantidad de bytes recibidos.

4.2 Arquitectura de la propuesta

En esta sección se describe la propuesta de *arquitectura global* (*multi-tier*) para el manejo de las soluciones agregadas al sistema de supervisión propuesto. Así como una *arquitectura específica* (también *multi-tier*) que se aplica a nivel de solución individual para el manejo del estado de esa solución. Ambas arquitecturas, la global y la específica, tienen como base la arquitectura de tres capas: acceso, procesamiento y datos. Mediante esas tres capas, de manera general se puede tener una entrada y una salida (capa de acceso), se puede realizar una transformación a los datos obtenidos (capa de procesamiento) y se puede tener un espacio para obtener dicha información (capa de datos).

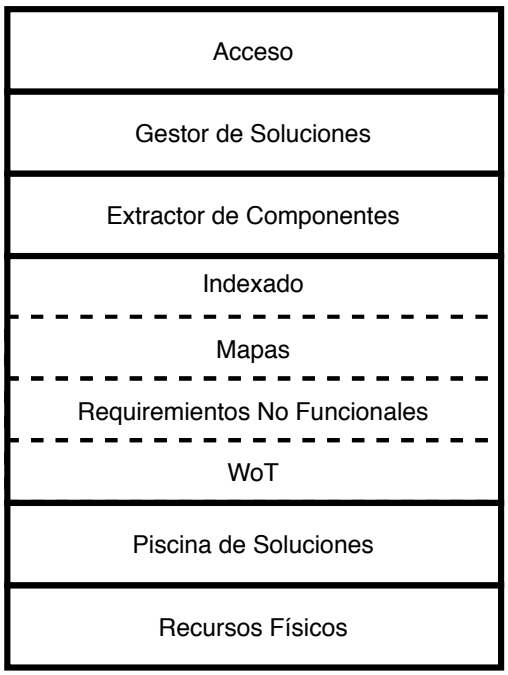


Figura 4.5: Arquitectura global del método propuesto.

Una arquitectura se conforma de capas primarias y capas secundarias. Una capa primaria se representa con un rectángulo de línea continua, mientras que una capa secundaria se encuentra dentro de una capa primaria y se representa con un rectángulo de línea punteada. La Figura 4.5 muestra la arquitectura global del método propuesto. Ésta opera en un nivel superior donde se encarga del

manejo de las soluciones. Consta de 6 capas primarias, las cuales se describen a continuación.

1. Acceso: recibe como entrada un usuario y el archivo YAML de la solución que se desea supervisar. Con estos datos se encarga de conceder o denegar el acceso al usuario proporcionado. Un archivo YAML, sirve para definir todos los componentes (contenedores virtuales) que corresponden a una solución, así como las configuraciones requeridas. Es mediante este archivo que se pueden desplegar múltiples contenedores.

2. Gestor de Soluciones: recibe como entrada el archivo YAML del usuario validado. Este se encarga de validar si el archivo recibido tiene un formato correcto.

3. Extractor de Componentes: recibe el archivo YAML validado. Interpreta el archivo YAML, extrae y valida los contenedores definidos en este archivo.

4. Indexado: recibe el listado de contenedores de la solución y, los procesa generando los mapas de estado. Esto se realiza de acuerdo al modelo definido anteriormente en el apartado del módulo de Indexado. Para fines de la propuesta, en esta capa se generan mapas adicionales para los requerimientos no funcionales y para el estándar de WoT.

5. Piscina de Soluciones: recibe los mapas de la capa anterior y un identificador único de la solución y. Estos datos son colocados en un espacio de común acceso conocido como piscina, donde se encuentran las soluciones que se pueden ejecutar en el sistema.

6. Recursos Físicos: esta capa representa el vínculo que se tiene entre las soluciones de la piscina y los recursos de entorno físico, como el medio para acceder a ellos.

Figura 4.6: Arquitectura específica del método propuesto.

La Figura 4.6 muestra la arquitectura específica del método propuesto. Ésta opera en un nivel inferior, donde se encarga de las actividades relacionadas a una solución individual. Consta de 8 capas primarias, las cuales se describen a continuación.

1. Acceso: recibe como entrada un usuario y el identificador de una solución. Esta capa se encarga de permitir el acceso por *front-end* (mediante una interfaz gráfica) y por *back-end* (sin una interfaz visual, por ejemplo, una API).

2. Gestor de Supervisión: recibe el identificador de solución proporcionado por un usuario. En esta capa se valida la existencia de dicha solución y. Solo si pasa la validación se continua a la siguiente capa.

3. Gestor Global de Soluciones: recibe el identificador de solución y validado. Posteriormente, hace uso del servicio de búsqueda global para obtener la estructura de mapas que permitirá obtener los valores.

4. Monitoreo: recibe la estructura de mapas de una solución y. Interpreta los mapas y hace uso del extractor de métricas para obtener los valores de cada métrica por cada recurso y por cada contenedor de dicha solución.

5. Indexado de valores: recibe los mapas con valores de la solución y. Posteriormente, realiza el guardado de estos mapas en el módulo de Indexado.

6. Diagnóstico: en esta capa se realiza una consulta para obtener los mapas que representan inicialmente una solución y junto con sus valores de acuerdo a un criterio de búsqueda, ya sea por tiempo o por lote. De los datos obtenidos se calculan agregados, los cuales son pasados por una verificación de umbrales. Esto genera el estado mediante una escala de calor, donde verde significa que todo ejecuta de manera holgada y rojo significa que se ha llegado al límite de la capacidad de la infraestructura..

7. Representación de Estado: recibe los mapas de una solución y y su estado mediante una escala de calor. Con lo que genera una representación y sus datos.

8. Re-Indexado de Datos: recibe los datos de la representación validados. Posteriormente los envía para su almacenamiento en el módulo de Indexado.

4.3 Implementación de un prototipo funcional basado en el método propuesto

En este apartado se describe la implementación del prototipo del método propuesto. El prototipo está basado en el diseño mostrado en la sección 4.1, lo que corresponde a cada uno de los módulos y las dos arquitecturas de la sección 4.2.

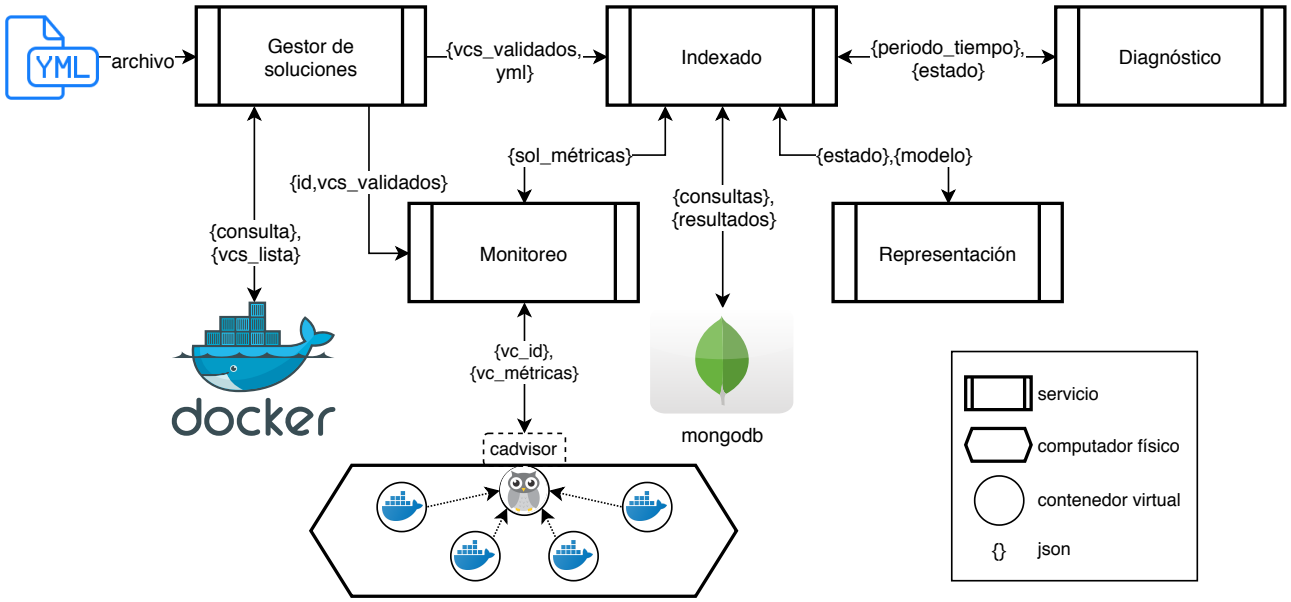


Figura 4.7: Diagrama de implementación del prototipo.

La Figura 4.7 muestra el diagrama de los módulos implementados en el prototipo. Cada uno de estos módulos fue implementado como un servicio contenerizado (dentro de un contenedor virtual) en lenguaje Python 3, los cuales se comunican mediante peticiones REST. Se asume que la solución que se agregue para ser supervisada ya se encuentra en ejecución. Esto debido a que el prototipo no se enfoca en el despliegue de soluciones, sino en su supervisión.

El Gestor de Soluciones es el encargado de leer un archivo YAML, a partir del cual genera un listado de contenedores. Posteriormente, estos contenedores son validados con el servidor *Docker*, de lo cual se obtiene una lista de contenedores activos. Después, el listado de contenedores validados, así como el YAML en formato JSON, son enviados al módulo de Indexado. De igual manera, se envía el *id* de dicha solución junto con una señal de iniciar el monitoreo de la solución proporcionada al módulo de Monitoreo.

El módulo de Monitoreo es un servicio que realiza la adquisición de métricas mediante un agente externo, llamado *cAdvisor*. cAdvisor es un API que provee tanto métricas de contenedores, como del equipo físico sobre el que se ejecuta. En este módulo también se realiza discretización de métricas ya que se puede configurar el intervalo de extracción de métricas en tiempos menores a 60 segundos entre

cada intervalo. Por intervalo (IT) se refiere a rangos de tiempo pequeños, en escala de segundos. Esto es $IT = \{1, 2, 3 \dots ts\}$ donde $(ts > 0 \wedge ts < 60)$. Por lo tanto, cada IT segundos este servicio envía una petición al módulo de Indexado con las métricas obtenidas de todos los contenedores de una solución y. Las métricas pueden ser reemplazadas por los factores de utilización al calcularlos con las fórmulas mostradas en el apartado de diseño.

En el módulo de Indexado se reciben las peticiones de los módulos: Gestor de Soluciones, Monitoreo, Representación y Diagnóstico. Cada uno con información específica de su rol. Este módulo se encarga de interactuar con la base de datos mediante operaciones de consulta, agregación, actualización y creación. En este caso la base de datos utilizada es MongoDB.

El módulo Representación genera la representación de los contenedores tomando en cuenta las dependencias definidas en el archivo YAML. Para obtener la información de dicho archivo se realiza una consulta al módulo de Indexado, el cual tiene acceso a esta información desde que es agregada una nueva solución.

En el módulo de Diagnóstico se define un estado por período de consulta. Esto se refiere al tamaño de ventana. La información correspondiente a este período es obtenida mediante una consulta al módulo de Indexado. Una vez obtenida dicha información se obtienen los agregados y se pasan por una verificación de umbrales definiendo su estado para ese período de tiempo.

5

Evaluación experimental y resultados

En este capítulo se describe la metodología de evaluación para conducir la evaluación experimental del prototipo descrito en el capítulo 4. La sección 5.3 muestra el listado de materiales utilizados en la evaluación controlada. La sección 5.4 corresponde a la descripción de las métricas empleadas. Por otra parte, en la sección 5.5 se describen las configuraciones probadas con el prototipo. La sección 5.6 se describen y presentan los resultados de la evaluación controlada. Finalmente, en la sección 5.7 se presenta el prototipo de supervisión aplicado en un estudio de caso.

5.1 Metodología de evaluación

El tipo de evaluación del método generado es experimental de prototipo. Donde el prototipo son un conjunto de servicios realizados para resolver el problema de supervisión de soluciones basadas en contenedores virtuales. La metodología se divide en dos partes, una evaluación controlada y un caso de estudio, como se muestra a continuación:

- *Evaluación controlada*: es un conjunto de experimentos que están dados por los aspectos críticos de prototipo, mismos que son definidos acorde a las características relevantes en la resolución del problema de supervisión. Estos son la cantidad de recursos/métricas a indexar, así como sus intervalos de recolección, los costos involucrados en la indexación, diagnóstico y en la representación. También se consideran las técnicas de recolección de métricas como entrega y extracción. El propósito de esta evaluación es entender la importancia de cada uno de los aspectos críticos y la influencia que tienen para permitir la comprobación de la hipótesis de este trabajo de tesis. Con esta parte se identifican uno o más aspectos de interés, mismos que son utilizados en un estudio de caso para una posterior evaluación. Un aspecto de interés identificado es el tiempo de servicio que toma realizar la representación de un sistema.

- *Estudio de caso*: sirve para comprobar la factibilidad del método de supervisión propuesto. Para ello se utiliza un conjunto de contenedores que forman parte de una solución real utilizando contenedores virtuales, éstos están definidos en un archivo YAML. Con el conjunto de contenedores se procede a desplegar escenarios que permiten evaluar la factibilidad del método de supervisión con respecto a los aspectos identificados en la evaluación controlada.

5.2 Definiciones

A continuación se definen algunos conceptos que son utilizados a lo largo de este capítulo.

- Tiempo de muestra: es el lapso de tiempo que existe entre un valor de las métricas (CPU, Memoria, Sistema de archivos y Red) y otro valor de las mismas.

- Tiempo de estado: es el lapso de tiempo entre cada obtención del estado de una solución.

- Tiempo de ventana: es el lapso de tiempo total contemplado en una solución, a partir del cual se pueden obtener uno o mas estados (tiempos de estado).

- Trabajador: es una entidad que se encarga de realizar una o varias tareas especificas, por lo cual tiene unas entradas definidas, realiza algún procesamiento y entrega los resultados en la ubicación que se le indique (o en la que tenga configurada por defecto), el cual usualmente es clonado para proveer paralelismo de tareas.

- Maestro/Trabajador: es un patrón de diseño de soluciones de software, donde el patrón es el encargado de recibir una petición junto con las entradas y repartirlas entre los trabajadores. Un trabajador realiza el procesamiento tomando en cuenta las tareas que le fueron asignadas y entrega el resultante acorde a sus configuraciones.

- Bloque de tiempo: también llamado bloque (no ha de confundirse con los bloques generados por la dispersión de IDA). Es utilizado en la sección 5.6.3 para referirse a los bloques obtenidos al dividir un tiempo de ventana.

- Factor de utilización: es un valor entre 0 y 1 que representa el grado de uso de un recurso (CPU, Memoria, Sistema de archivos y Red), también es mencionado como utilización.

- Escala de riesgo: es el nivel de carga (bajo, medio o alto) en el que se encuentra un factor de utilización.

5.3 Lista de materiales

El listado de materiales se divide en dos categorías: hardware (se refiere a la infraestructura física sobre la que se ejecutaron tanto el prototipo como las soluciones de software evaluadas) y software (referente a los programas que ayudaron en el desarrollo y despliegue del prototipo). Estos materiales son descritos a continuación.

Equipo	Cores	RAM	HDD	OS
Disys1	6	12 GB	256 GB	CentOS 7
Disys3	6	12 GB	256 GB	CentOS 7
compute11	12	64 GB	5.4 TB	CentOS 7

Tabla 5.1: Infraestructura del clúster.

5.3.1 Hardware

Para el despliegue y evaluación del prototipo se usó la infraestructura de un clúster mostrada en la Tabla 5.1. Esta infraestructura se encuentra ubicada en el Cinvestav Tamaulipas.

5.3.2 Software

- **Docker:** es un proyecto de código libre que, mediante virtualización ligera, permite encapsular aplicaciones y sus dependencias en contenedores virtuales. Un contenedor se crea a partir de una imagen de un contenedor y a su vez esta imagen es creada por un archivo *Dockerfile*. Con Docker se pueden desplegar y controlar múltiples contenedores virtuales en un equipo con la tecnología *Docker-compose* y escalarse a múltiples equipos mediante *Docker swarm*, ambos usando archivos YAML.

- **cAdvisor:** es un demonio de código libre desarrollado por Google que provee métricas respecto al uso y rendimiento de contenedores virtuales en ejecución. También permite extraer métricas del equipo físico sobre el que se está ejecutando.

- **MongoDB:** es una base de datos no relacional basada en documentos, que guarda información en objetos JSON. Esto permite trabajar de manera natural con múltiples dimensiones en los datos.

- ***The Information Dispersal Algorithm* (IDA)** [46]: es un algoritmo que se utiliza para separar bloques de datos en piezas agregándoles bytes extras de información y así evitar

que éstos puedan ser entendidos de manera independiente por algún lector no deseado. Este algoritmo divide un archivo F en n piezas, con posibilidad de recuperar el contenido original con m piezas cualesquiera. Esto sucede siempre que $m < n$ y pueden ocurrir k errores en $n - m$ piezas.

- **Kulla** [48]: es un modelo de construcción a bloques para crear aplicaciones distribuidas y/o paralelas. Esto mediante aplicaciones en unidades *Blocks*, que se pueden encadenar en *Bricks* y aplicando patrones recursivos forman *Boxes*. Ejemplos de Bricks son *Divide&Containerize* (paralelismo de datos), *Pipe&Blocks* (streaming) y *Manager/Block* (paralelismo de tareas).

5.3.3 Conjunto de datos

El conjunto de datos utilizado (Tabla 5.2) consiste de archivos YAML que representan soluciones de software comunes haciendo uso de contenedores virtuales. Se usan archivos YAML debido a que los archivos con este formato son utilizados para definir todos los componentes de una solución, así como los parámetros de configuración, mismos que son interpretados por una plataforma de contenedores permitiendo su despliegue. Estos trabajos fueron desarrollados para ejecutarse individualmente e implementando patrones de flujos de trabajo. Las soluciones con ids 4-7 implementan una cantidad diferente de soluciones IDA (1,10,10 y 100 respectivamente), todas con una configuración de $n = 8$ y $m = 4$.

5.4 Métricas

Las métricas que se definieron para medir el rendimiento del prototipo que implementa el sistema de supervisión son las siguientes:

- Tiempo de servicio (ST): representa el tiempo requerido por un módulo para completar una tarea dada.

Id	Solución	Contenedor
1	smallapp	bubble
2	ida	ida
3	kulla	k_masterslave
4	sol1c1	ida (x1)
5	sol1c10	ida (x10)
6	sol10c1	ida (x10)
7	sol10c10	ida (x100)

Tabla 5.2: Contenedores de las soluciones utilizadas en la experimentación.

- Tiempo de respuesta (RT): representa el tiempo observado por un usuario final desde que se envía una solicitud a un servicio, hasta que ésta es completada.

5.5 Configuraciones

En la Figura 5.1 se muestra el diseño experimental correspondiente a la evaluación controlada y al caso de estudio. En la evaluación controlada se realizaron dos experimentos, uno con respecto a la detección de cambios de estado en los datos obtenidos del monitoreo y otro variando la cantidad de soluciones como de contenedores por solución. Cada experimento se repitió 31 veces, con cada una de las configuraciones probadas. El primer experimento consiste en mostrar la capacidad del prototipo de capturar el estado de una solución de acuerdo a la información disponible, por eso se varían tres tiempos, ya que estos influyen en la cantidad de información que se disponga. En cada experimento se analiza cuál es la mejor configuración de acuerdo a las necesidades. En el experimento de incremento de soluciones/contenedores se calculan los tiempos se servicio de cada una de las cuatro etapas. También el tiempo de respuesta, que abarca el tiempo que toma realizar todas las etapas desde la primera hasta la última.

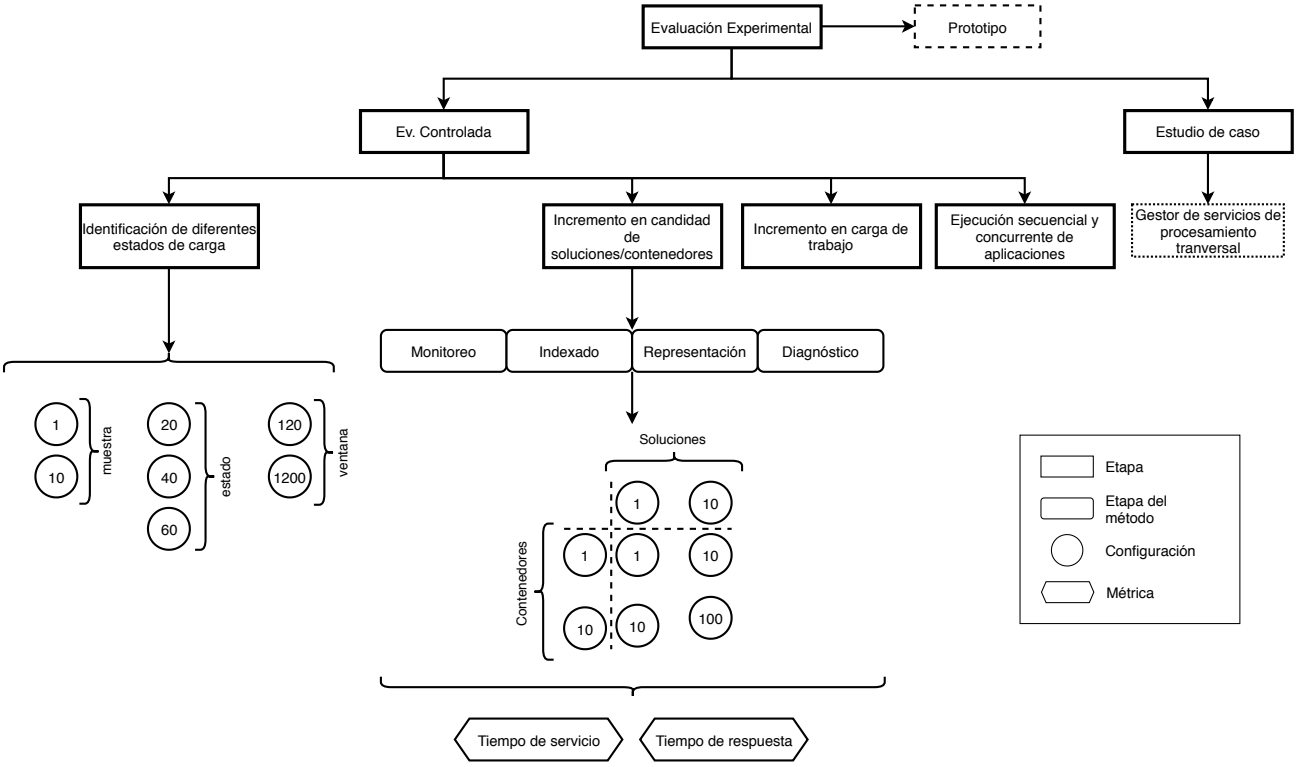


Figura 5.1: Diagrama de configuraciones probadas en la experimentación controlada.

5.6 Fase de evaluación controlada

Este experimento tiene como propósito evaluar el prototipo desarrollado, identificando aspectos de interés con respecto a la configuración de tiempos para la captura métricas del monitoreo, así como para generar el estado, tomando en cuenta los tiempos de servicio por cada una de las etapas y los tiempos de respuesta desde la carga de una nueva solución hasta obtener el diagnóstico de la misma.

5.6.1 Identificación de diferentes estados de carga

Este experimento consiste en demostrar que el prototipo es capaz de capturar e identificar las variaciones en el nivel de carga de una solución, esto de acuerdo a los umbrales previamente definidos en el Capítulo 4. Se utilizaron las soluciones mostradas en la Tabla 5.2, las cuales se ejecutaron sobre

la infraestructura descrita en la Tabla 5.1. A dichas soluciones se les aplicaron tres variaciones de tiempo, definidas como: tiempo de muestra, tiempo de estado y tiempo de ventana, resumidas en la Tabla 5.3. El tiempo de muestra se refiere al tiempo que transcurre para obtener nuevamente valores de monitoreo. El tiempo de estado y de ventana están relacionados ya que si se tiene un tiempo de ventana de x y un tiempo de estado de y, la ventana consiste en dividir este tiempo en secciones de tiempo y. Para cada configuración se consideran cuatro recursos: CPU, memoria, sistema de archivos y red.

Para cada solución se muestran sus gráficas y datos estadísticos de cada configuración. También se muestra la escala de riesgo obtenida en cada estado y el porcentaje de datos entre 1, 2 y 3 desviaciones estándar (sigma) de la media, de acuerdo a la regla empírica de la distribución normal. Esta regla establece que para que los valores de los datos sigan una distribución normal, el porcentaje de valores que se encuentran dentro de uno, dos y tres sigmas son: 68.27%, 95.45% y 99.73% respectivamente. Las gráficas muestran un punto de agregación por cada estado, este es la mediana y el error (+- desviación estándar).

ventana	120						1200					
estado	20		40		60		20		40		60	
muestra	1	10	1	10	1	10	1	10	1	10	1	10

Tabla 5.3: Configuraciones del experimento variando tiempos de ventana, estado y muestra.

5.6.1.1. Smallapp

La primera solución contiene una aplicación llamada *Smallapp*, la cual consiste en ordenar un conjunto de números aleatorios de longitud $n = 10,000$ mediante el método de ordenamiento burbuja. Esta misma aplicación se encarga de generar los números aleatorios entre un rango $[1, n]$. Una vez generados los datos se procede a ordenarlos y posteriormente imprimirlos. Una vez finalizado este proceso se repite nuevamente para extender el tiempo necesario para completar el experimento.

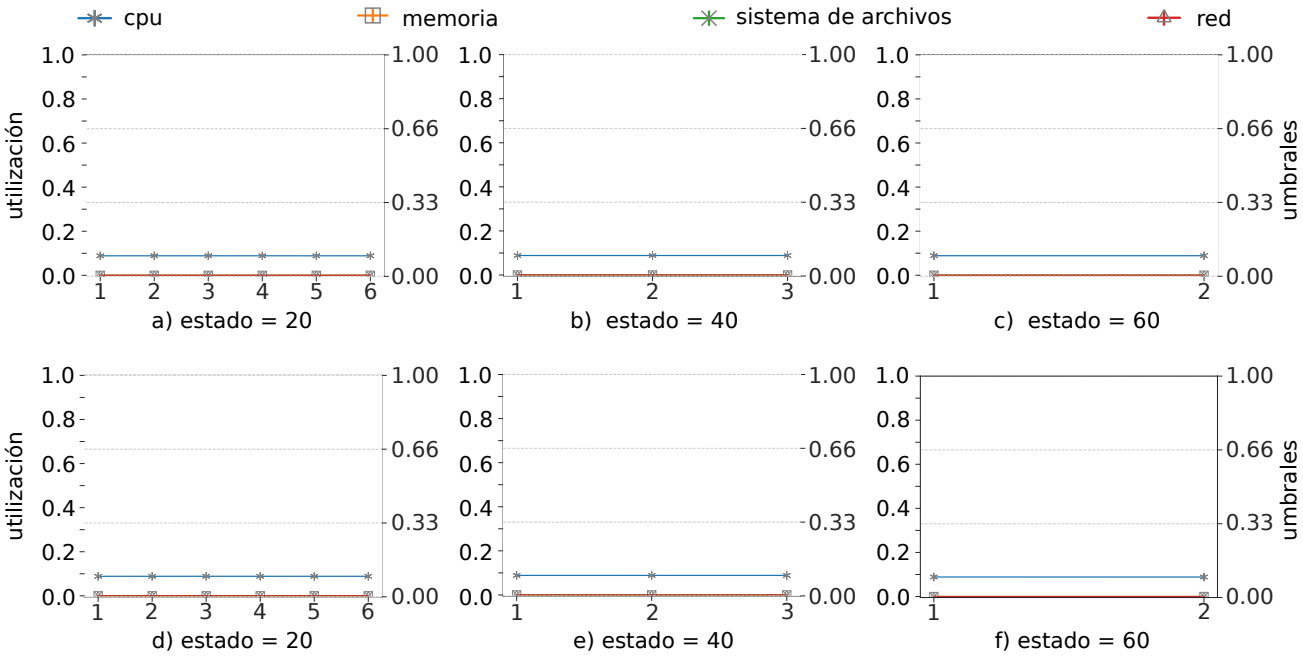


Figura 5.2: Configuraciones de estado y muestra para la solución *Smallapp* con ventana = 120.

La Figura 5.2 muestra los resultados de estado con la configuración de ventana = 120 segundos. En el eje vertical izquierdo se muestra la utilización. En el eje vertical derecho se muestran las escalas de riesgo (bajo, medio y alto). El eje horizontal corresponde a los estados obtenidos. Los incisos a, b y c corresponden a la muestra = 1, lo que varía es el tiempo de estado 20, 40 y 60 segundos. Los incisos d, e y f son equivalentes a sus respectivas partes a, b y c, pero con un tiempo de muestra = 10 segundos. Estas configuraciones se refieren a un tiempo de 10 segundos al obtener las muestras, por lo que cada 10 segundos se tendría una muestra. Donde esa muestra no representa un valor resumido sino un valor único en esos 10 segundos. Esta variación en la cantidad de datos obtenidos permite seguir conservando el comportamiento original representando el mismo estado. Se puede observar cómo se reduce la cantidad de estados conforme se aumenta el tiempo entre cada obtención de ellos. Sin embargo, el comportamiento se conserva ya que este no sufre gran variación. Esto se puede comprobar en la Tabla 5.4, donde se muestran los datos estadísticos de la Figura 5.2 incisos a, b y c, y la Tabla 5.5 que muestra los datos estadísticos de la Figura 5.2 incisos d, e y f.

Estadístico	ventana = 120,muestra = 1											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	186	93	62	186	93	62	186	93	62	186	93	62
min	0.0832	0.0840	0.0843	9.57E-06	9.57E-06	9.57E-06	8.85E-05	9.13E-05	9.13E-05	0	0	0
max	0.0950	0.0944	0.0934	9.57E-06	9.57E-06	9.57E-06	0.0003	0.0003	0.0003	0	0	0
media	0.0893	0.0893	0.0893	9.57E-06	9.57E-06	9.57E-06	0.0002	0.0002	0.0002	0	0	0
mediana	0.0886	0.0886	0.0886	9.57E-06	9.57E-06	9.57E-06	0.0002	0.0002	0.0002	0	0	0
std	0.0017	0.0017	0.0017	0	0	0	5.79E-05	5.81E-05	5.82E-05	0	0	0
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	144	72	47	186	93	62	111	56	36	186	93	62
sigma1_perc	77.42	77.42	75.81	100.00	100.00	100.00	59.68	60.22	58.06	100.00	100.00	100.00
sigma2_n	173	87	57	186	93	62	186	93	62	186	93	62
sigma2_perc	93.01	93.55	91.94	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
sigma3_n	184	92	62	186	93	62	186	93	62	186	93	62
sigma3_perc	98.92	98.92	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Tabla 5.4: Estadísticos descriptivos del experimento con *Smallapp* con ventana = 120 y muestra = 1.

Estadístico	ventana = 120, muestra = 10											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	186	93	62	186	93	62	186	93	62	186	93	62
min	0.0835	0.0863	0.0848	9.57E-06	9.57E-06	9.57E-06	8.85E-05	8.99E-05	9.13E-05	0	0	0
max	0.0950	0.0936	0.0933	9.57E-06	9.57E-06	9.57E-06	0.0003	0.0003	0.0003	0	0	0
media	0.0893	0.0893	0.0893	9.57E-06	9.57E-06	9.57E-06	0.0002	0.0002	0.0002	0	0	0
mediana	0.0886	0.0886	0.0886	9.57E-06	9.57E-06	9.57E-06	0.0002	0.0002	0.0002	0	0	0
std	0.0017	0.0016	0.0017	0	0	0	5.79E-05	5.81E-05	5.82E-05	0	0	0
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	144	71	47	186	93	62	111	55	36	186	93	62
sigma1_perc	77.42	76.34	75.81	100.00	100.00	100.00	59.68	59.14	58.06	100.00	100.00	100.00
sigma2_n	174	88	58	186	93	62	186	93	62	186	93	62
sigma2_perc	93.55	94.62	93.55	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
sigma3_n	183	93	62	186	93	62	186	93	62	186	93	62
sigma3_perc	98.39	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Tabla 5.5: Estadísticos descriptivos del experimento con *Smallapp* con ventana = 120 y muestra = 10.

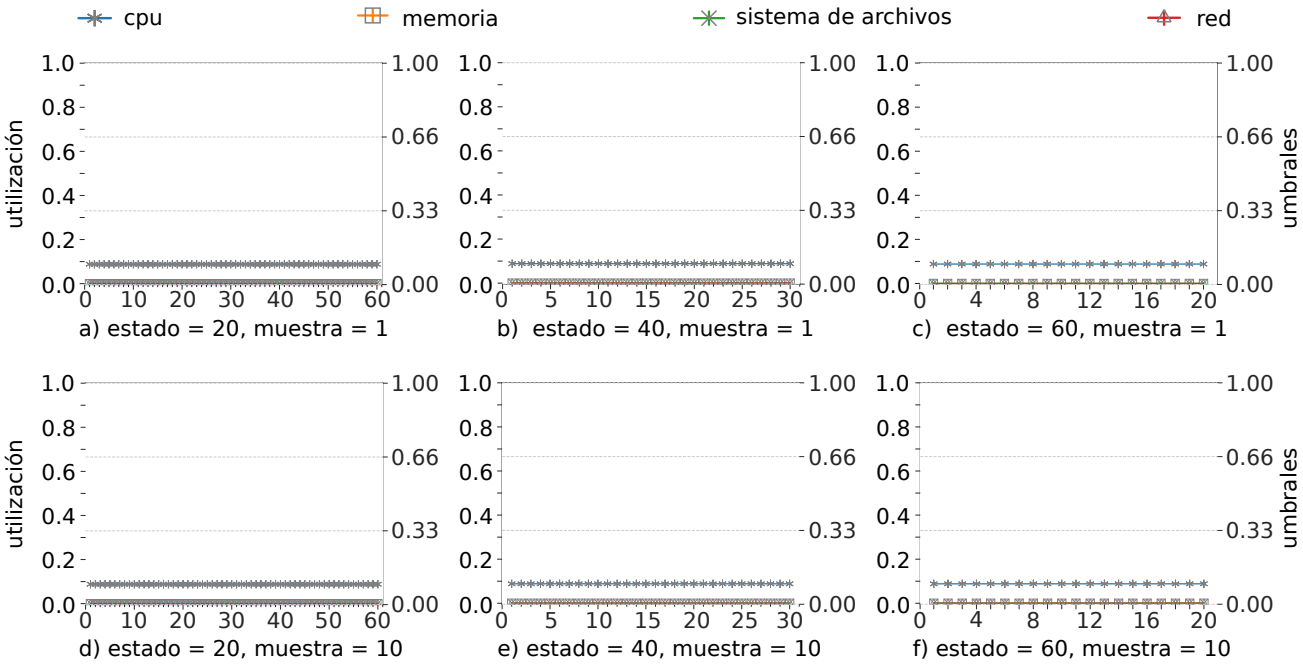


Figura 5.3: Configuraciones de estado y muestra para la solución *Smallapp* con ventana = 1200.

La Figura 5.3 muestra los resultados de estado con la configuración de ventana = 1200 segundos. Se conservan los mismos ejes en las gráficas de este experimento como en el anterior. Por lo que el eje vertical izquierdo son utilizaciones, mientras que el vertical derecho son las escalas de riesgo y el horizontal los estados obtenidos. Se tiene una mayor cantidad de muestras debido a que se trata de una ventana de mayor tamaño. Por lo que se puede confirmar que el comportamiento mostrado con una ventana pequeña se extiende a un tiempo de ventana mayor. Se muestra cómo el aumento en la ventana y la variación en la frecuencia también conserva el comportamiento original. La Tabla 5.6 muestra los datos estadísticos de la Figura 5.4 incisos a, b y c. La Tabla 5.7 muestra los datos estadísticos de la Figura 5.4 incisos d, e y f.

Estadístico	ventana = 1200, muestra = 1											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1860	930	620	1860	930	620	1860	930	620	1860	930	620
min	0.0832	0.0840	0.0843	9.57E-06	9.57E-06	9.57E-06	8.85E-05	9.13E-05	9.13E-05	0	0	0
max	0.0950	0.0944	0.0934	9.57E-06	9.57E-06	9.57E-06	0.0021	0.0021	0.0021	0	0	0
media	0.0883	0.0883	0.0883	9.57E-06	9.57E-06	9.57E-06	0.0011	0.0011	0.0011	0	0	0
mediana	0.0882	0.0882	0.0882	9.57E-06	9.57E-06	9.57E-06	0.0011	0.0011	0.0011	0	0	0
std	0.0007	0.0007	0.0007	0	0	0	0.0006	0.0006	0.0006	0	0	0
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	1792	896	597	1860	930	620	1071	537	358	1860	930	620
sigma1_perc	96.34	96.34	96.29	100.00	100.00	100.00	57.58	57.74	57.74	100.00	100.00	100.00
sigma2_n	1802	902	600	1860	930	620	1860	930	620	1860	930	620
sigma2_perc	96.88	96.99	96.77	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
sigma3_n	1809	905	603	1860	930	620	1860	930	620	1860	930	620
sigma3_perc	97.26	97.31	97.26	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Tabla 5.6: Estadísticos descriptivos del experimento con *Smallapp* con ventana = 1200 y muestra = 1.

Estadístico	ventana = 1200, muestra = 10											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1860	930	620	1860	930	620	1860	930	620	1860	930	620
min	0.0835	0.0863	0.0848	9.57E-06	9.57E-06	9.57E-06	8.85E-05	8.99E-05	9.13E-05	0	0	0
max	0.0950	0.0936	0.0933	9.57E-06	9.57E-06	9.57E-06	0.0021	0.0021	0.0021	0	0	0
media	0.0883	0.0883	0.0883	9.57E-06	9.57E-06	9.57E-06	0.0011	0.0011	0.0011	0	0	0
mediana	0.0882	0.0882	0.0882	9.57E-06	9.57E-06	9.57E-06	0.0011	0.0011	0.0011	0	0	0
std	0.0007	0.0006	0.0006	0	0	0	0.0006	0.0006	0.0006	0	0	0
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	1791	895	597	1860	930	620	1071	537	358	1860	930	620
sigma1_perc	96.29	96.24	96.29	100.00	100.00	100.00	57.58	57.74	57.74	100.00	100.00	100.00
sigma2_n	1802	900	600	1860	930	620	1860	930	620	1860	930	620
sigma2_perc	96.88	96.77	96.77	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
sigma3_n	1810	904	602	1860	930	620	1860	930	620	1860	930	620
sigma3_perc	97.31	97.20	97.10	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Tabla 5.7: Estadísticos descriptivos del experimento con *Smallapp* con ventana = 1200 y muestra = 10.

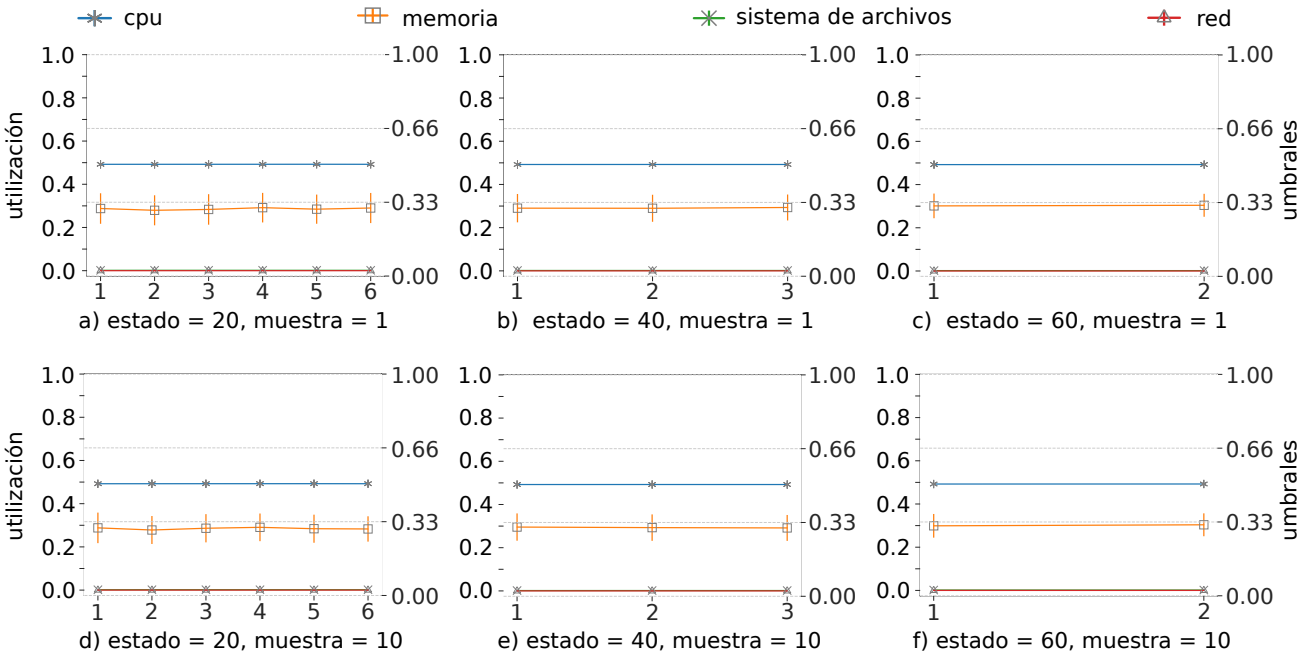


Figura 5.4: Configuraciones de estado y muestra para la solución IDA con ventana = 120.

5.6.1.2. IDA

Esta solución tiene un contenedor que implementa el algoritmo de dispersión IDA. Este algoritmo realiza la Dispersión y Recuperación de archivos de 500MB de tal forma que simula contar con 3 trabajadores cada uno realizando la Dispersión y Recuperación de un archivo, lo que produce una ejecución paralela. Una vez finalizado dicho proceso en el trabajador, se vuelve a repetir.

La Figura 5.4 muestra los resultados de estado con la configuración de ventana = 120 segundos. El eje vertical izquierdo corresponde a la utilización. El eje vertical derecho contiene las escalas de riesgo. El eje horizontal corresponde a los estados. En este experimento se aplica un filtro al obtener las muestras, ya que obtiene una muestra cada 10 segundos. Se aprecia un estado estabilizado en CPU de $1/2$, mientras que la memoria tiene un uso aproximado de $1/3$, el cual se conserva en las tres variaciones de tiempo de estado. La Tabla 5.8 muestra los datos estadísticos de los incisos a, b y c, y la Tabla 5.9 de los incisos d, e y f.

Estadístico	ventana = 120, muestra = 1											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	186	93	62	186	93	62	186	93	62	186	93	62
min	0.4620	0.4647	0.4667	0.1109	0.118012824	0.1309	2.00E-08	0.0007	0.0007	0	0	0
max	0.4935	0.4935	0.4934	0.3673	0.348638451	0.3300	0.0023	0.0023	0.0023	0	0	0
media	0.4907	0.4906	0.4906	0.2732	0.276154576	0.2725	0.0016	0.0016	0.0016	0	0	0
mediana	0.4925	0.4924	0.4925	0.2858	0.290222439	0.3015	0.0017	0.0017	0.0017	0	0	0
std	0.0054	0.0053	0.0054	0.0685	0.062129869	0.0548	0.0004	0.0004	0.0004	0	0	0
escala de riesgo	medio	medio	medio	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	168	84	55	110	57	40	122	57	39	186	93	62
sigma1_perc	90.32	90.32	88.71	59.14	61.29032258	64.52	65.59	61.29	62.90	100.00	100.00	100.00
sigma2_n	175	88	58	184	92	61	182	89	60	186	93	62
sigma2_perc	94.09	94.62	93.55	98.92	98.92473118	98.39	97.85	95.70	96.77	100.00	100.00	100.00
sigma3_n	180	90	60	186	93	62	185	93	62	186	93	62
sigma3_perc	96.77	96.77	96.77	100.00	100	100.00	99.46	100.00	100.00	100.00	100.00	100.00

Tabla 5.8: Estadísticos descriptivos del experimento con IDA con ventana = 120 y muestra = 1.

Estadístico	ventana = 120, muestra = 10											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	186	93	62	186	93	62	186	93	62	186	93	62
min	0.4607	0.4636	0.4659	0.1017	0.1149	0.1254	2.00E-08	0.0004	0.0007	0	0	0
max	0.4935	0.4935	0.4935	0.3701	0.3551	0.3376	0.0023	0.0023	0.0023	0	0	0
media	0.4907	0.4906	0.4907	0.2741	0.2761	0.2745	0.0016	0.0016	0.0016	0	0	0
mediana	0.4925	0.4925	0.4925	0.2848	0.2929	0.3022	0.0017	0.0017	0.0017	0	0	0
std	0.0053	0.0054	0.0051	0.0640	0.0608	0.0535	0.0004	0.0004	0.0004	0	0	0
escala de riesgo	medio	medio	medio	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	168	84	55	118	57	41	122	61	39	186	93	62
sigma1_perc	90.32	90.32	88.71	63.44	61.29	66.13	65.59	65.59	62.90	100.00	100.00	100.00
sigma2_n	175	88	58	180	92	61	182	88	60	186	93	62
sigma2_perc	94.09	94.62	93.55	96.77	98.92	98.39	97.85	94.62	96.77	100.00	100.00	100.00
sigma3_n	179	90	60	186	93	62	185	92	62	186	93	62
sigma3_perc	96.24	96.77	96.77	100.00	100.00	100.00	99.46	98.92	100.00	100.00	100.00	100.00

Tabla 5.9: Estadísticos descriptivos del experimento con IDA con ventana = 120 y muestra = 10.

La Figura 5.5 muestra los resultados de estado con la configuración de ventana = 1200 segundos. Los incisos a, b y c corresponden a la muestra = 1, los incisos d, e y f corresponden a la muestra = 10. En ambos casos se varía el tiempo de estado en 20, 40 y 60 segundos. En este caso se tienen valores estabilizados en CPU de $1/2$, con una variación mínima. La memoria presenta una mayor variación que el CPU ya que su error (std) es mayor. El error disminuye conforme se incrementa el tiempo entre cada estado. La configuración de 10 segundos hace un filtrado en la cantidad de datos, ya que se tendría una muestra cada 10 segundos en lugar de 10 muestras. Este filtrado no produce un cambio significativo en el estado ya que se mantiene el mismo comportamiento en cada caso. La Tabla 5.10 muestra los datos estadísticos de la Figura 5.5 incisos a, b y c. La Tabla 5.11 muestra los datos estadísticos de la Figura 5.5 incisos d, e y f.

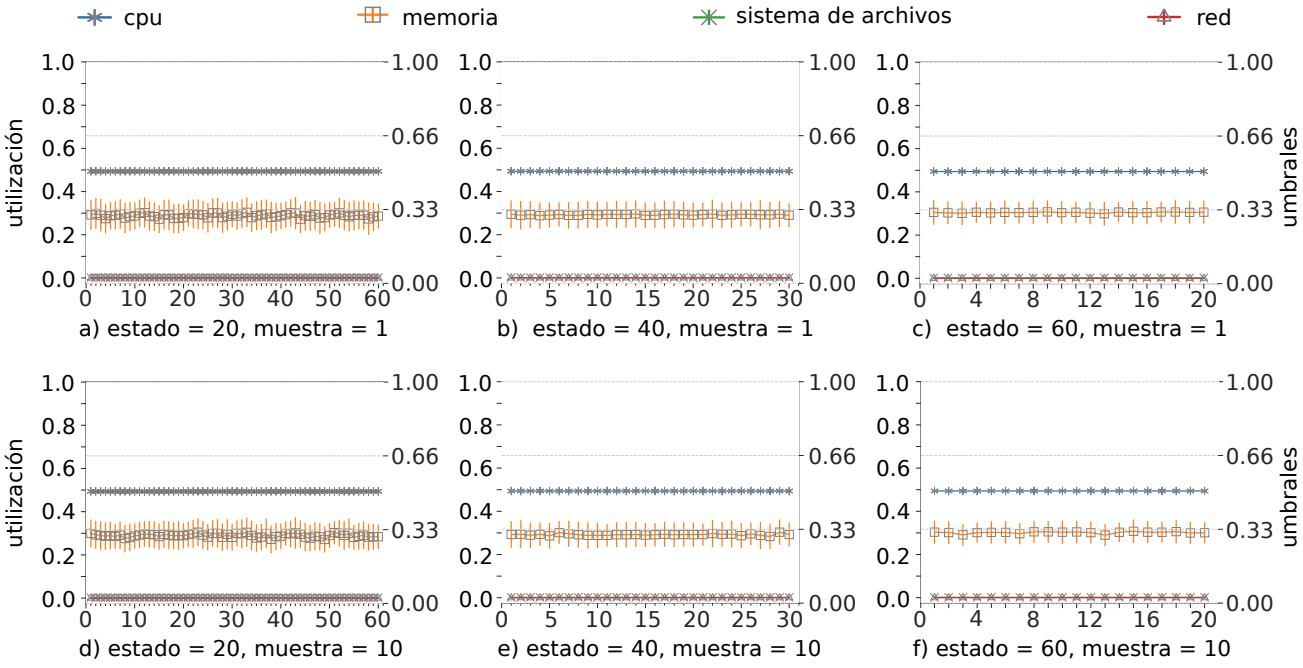


Figura 5.5: Configuraciones de estado y muestra para la solución IDA con ventana = 1200.

Estadístico	ventana = 1200, muestra = 1											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1860	930	620	1860	930	620	1860	930	620	1860	930	620
min	0.4620	0.4647	0.4667	0.1109	0.1180	0.1309	0.0000	0.0007	0.0007	0	0	0
max	0.4943	0.4943	0.4943	0.3673	0.3504	0.3340	0.0023	0.0023	0.0023	0	0	0
media	0.4937	0.4937	0.4937	0.2773	0.2780	0.2777	0.0017	0.0017	0.0017	0	0	0
mediana	0.4940	0.4940	0.4940	0.2887	0.2922	0.3043	0.0017	0.0017	0.0017	0	0	0
std	0.0020	0.0020	0.0020	0.0661	0.0597	0.0518	0.0004	0.0004	0.0004	0	0	0
escala de riesgo	medio	medio	medio	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	1815	908	605	1123	536	427	1158	577	386	1860	930	620
sigma1_perc	97.58	97.63	97.58	60.38	57.63	68.87	62.26	62.04	62.26	100.00	100.00	100.00
sigma2_n	1827	913	609	1858	929	619	1808	904	603	1860	930	620
sigma2_perc	98.23	98.17	98.23	99.89	99.89	99.84	97.20	97.20	97.26	100.00	100.00	100.00
sigma3_n	1836	917	612	1860	930	620	1859	930	620	1860	930	620
sigma3_perc	98.71	98.60	98.71	100.00	100.00	100.00	99.95	100.00	100.00	100.00	100.00	100.00

Tabla 5.10: Estadísticos descriptivos del experimento con IDA con ventana = 1200 y muestra = 1.

Estadístico	ventana = 1200, muestra = 10											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1860	930	620	1860	930	620	1860	930	620	1860	930	620
min	0.4607	0.4636	0.4659	0.1017	0.1149	0.1254	2.00E-08	0.0004	0.0007	0	0	0
max	0.4943	0.4943	0.4943	0.3709	0.3577	0.3417	0.0023	0.0023	0.0023	0	0	0
media	0.4937	0.4937	0.4937	0.2775	0.2781	0.2774	0.0017	0.0017	0.0017	0	0	0
mediana	0.4940	0.4940	0.4940	0.2894	0.2924	0.3025	0.0017	0.0017	0.0017	0	0	0
std	0.0020	0.0020	0.0019	0.0620	0.0571	0.0504	0.0004	0.0004	0.0004	0	0	0
escala de riesgo	medio	medio	medio	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	1815	908	605	1179	570	428	1158	606	386	1860	930	620
sigma1_perc	97.58	97.63	97.58	63.39	61.29	69.03	62.26	65.16	62.26	100.00	100.00	100.00
sigma2_n	1827	913	609	1791	910	619	1808	889	603	1860	930	620
sigma2_perc	98.23	98.17	98.23	96.29	97.85	99.84	97.20	95.59	97.26	100.00	100.00	100.00
sigma3_n	1837	917	611	1860	930	619	1859	929	620	1860	930	620
sigma3_perc	98.76	98.60	98.55	100.00	100.00	99.84	99.95	99.89	100.00	100.00	100.00	100.00

Tabla 5.11: Estadísticos descriptivos del experimento con IDA con ventana = 1200 y muestra = 10.

5.6.1.3. Kulla

Esta solución implementa el patrón Maestro/Trabajador utilizando el modelo de Kulla, el cual mediante estructuras de construcción permite crear otras aplicaciones y/o patrones, por lo que permite ejecutar un flujo a través de tres filtros: hash, compresión y codificación. Este flujo es aplicado sobre un conjunto de 32 archivos de 200MB utilizando 6 trabajadores. Una vez finalizado el flujo, éste se lanza nuevamente hasta cumplir con el tiempo de ventana deseado.

La Figura 5.6 muestra los resultados de estado con la configuración de ventana = 120 segundos. El eje vertical izquierdo corresponde a la utilización, el eje vertical derecho a las escalas de riesgo y el eje horizontal son los estados. Los incisos a, b y c corresponden a la muestra = 1. Los incisos d, e y f corresponden a la muestra = 10. En ambos casos se varía el tiempo de estado en 20, 40 y 60 segundos. Este caso tiene un tiempo de ventana pequeño, el cual muestra un estado de carga alto en memoria (mayor a 2/3). Dicho comportamiento se presenta en los casos de a, b y c, aunque se ve disminuido por la reducción de estados obtenidos de 6, 3 y 2 respectivamente. Aplicando la variación de frecuencia en la toma de muestras se reduce la cantidad de muestras de 10 a 1. Esta reducción en los datos produce un comportamiento semejante, pero con menos muestras. El estado para la memoria se mantiene en alto y para CPU en bajo. Este comportamiento se presenta de igual manera conforme se incrementa el tiempo entre cada estado, pero produce una reducción en la cantidad de estados obtenidos. La Tabla 5.12 muestra los datos estadísticos de la Figura 5.6 incisos a, b y c, y la Tabla 5.13 los incisos d, e y f.

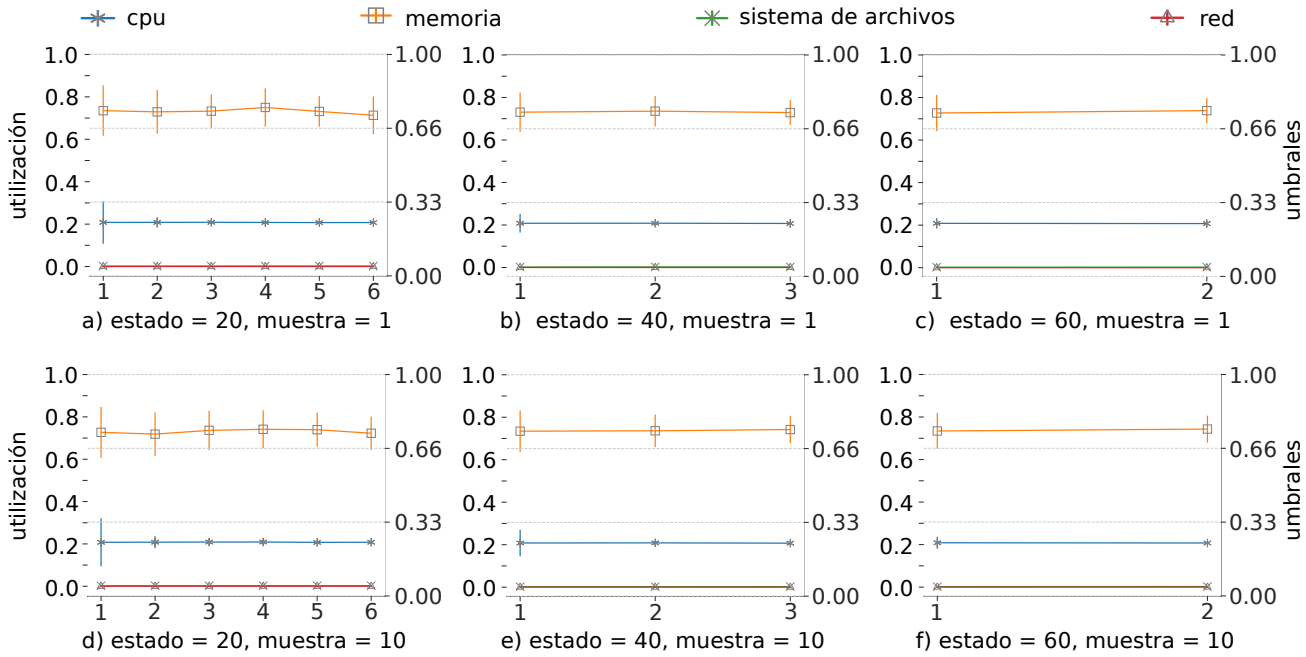


Figura 5.6: Configuraciones de estado y muestra para la solución Kulla con ventana = 120.

Estadístico	ventana = 120, muestra = 1											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	186	93	62	186	93	62	186	93	62	186	93	62
min	0.1950	0.1961	0.1983	0.2035	0.2819	0.3124	5.52E-08	5.52E-08	5.52E-08	2.11E-05	2.12E-05	2.12E-05
max	0.7688	0.4519	0.3430	0.7958	0.7790	0.7755	0.0043	0.0043	0.0043	0.0071	0.0039	0.0026
media	0.2132	0.2120	0.2113	0.7000	0.7136	0.7162	0.0023	0.0023	0.0023	0.0001	0.0001	0.0001
mediana	0.2081	0.2081	0.2082	0.7326	0.7313	0.7334	0.0026	0.0026	0.0026	4.21E-05	4.21E-05	4.21E-05
std	0.0424	0.0257	0.0174	0.0924	0.0746	0.0731	0.0014	0.0013	0.0014	0.0006	0.0004	0.0003
escala de riesgo	bajo	bajo	bajo	alto	alto	alto	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	183	91	61	159	86	58	114	58	38	181	90	59
sigma1_perc	98.39	97.85	98.39	85.48	92.47	93.55	61.29	62.37	61.29	97.31	96.77	95.16
sigma2_n	184	92	61	176	89	59	186	93	62	183	91	60
sigma2_perc	98.92	98.92	98.39	94.62	95.70	95.16	100.00	100.00	100.00	98.39	97.85	96.77
sigma3_n	184	92	61	182	90	60	186	93	62	184	92	61
sigma3_perc	98.92	98.92	98.39	97.85	96.77	96.77	100.00	100.00	100.00	98.92	98.92	98.39

Tabla 5.12: Estadísticos descriptivos del experimento con Kulla con ventana = 120 y muestra = 1.

Estadístico	ventana = 120, muestra = 10											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	186	93	62	186	93	62	186	93	62	186	93	62
min	0.1970	0.1972	0.1986	0.2307	0.2761	0.3286	5.52E-08	5.52E-08	5.52E-08	2.11E-05	2.12E-05	2.13E-05
max	0.8409	0.5591	0.3642	0.8088	0.7997	0.7820	0.0043	0.0043	0.0043	0.0192	0.0048	0.0029
media	0.2138	0.2133	0.2117	0.6982	0.7133	0.7170	0.0023	0.0023	0.0023	0.0002	0.0001	0.0001
mediana	0.2082	0.2082	0.2083	0.7304	0.7375	0.7393	0.0026	0.0024	0.0026	4.22E-05	4.22E-05	4.22E-05
std	0.0481	0.0367	0.0201	0.0940	0.0802	0.0746	0.0014	0.0012	0.0014	0.0014	0.0005	0.0004
escala de riesgo	bajo	bajo	bajo	alto	alto	alto	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	183	91	61	159	81	58	114	65	38	183	90	59
sigma1_perc	98.39	97.85	98.39	85.48	87.10	93.55	61.29	69.89	61.29	98.39	96.77	95.16
sigma2_n	184	92	61	178	89	59	186	93	62	185	91	61
sigma2_perc	98.92	98.92	98.39	95.70	95.70	95.16	100.00	100.00	100.00	99.46	97.85	98.39
sigma3_n	184	92	61	183	90	60	186	93	62	185	92	61
sigma3_perc	98.92	98.92	98.39	98.39	96.77	96.77	100.00	100.00	100.00	99.46	98.92	98.39

Tabla 5.13: Estadísticos descriptivos del experimento con Kulla con ventana = 120 y muestra = 10.

La Figura 5.7 muestra los resultados de estado con la configuración de ventana = 1200 segundos. El tiempo de ventana es representativo de una ventana de tamaño mayor, por lo que se extiende en el tiempo. Esto genera una mayor cantidad de datos para obtener el estado. Se pueden apreciar de mejor manera las variaciones entre cada estado. Las variaciones de estado son: alto para memoria y bajo para el CPU. Se tiene una reducción en la variabilidad de los estados conforme se incrementa el tiempo entre cada uno de ellos. La reducción de muestras de 10 a 1 resulta en un período mayor de tiempo con diez veces menos muestras. Esto produce que se obtenga un estado de memoria alto (mayor a 2/3) y un estado de CPU bajo (menor a 1/3), lo que conserva el comportamiento sin reducir muestras. La Tabla 5.14 muestra los datos estadísticos de la Figura 5.7 incisos a, b y c, y la Tabla 5.15 los incisos d, e y f.

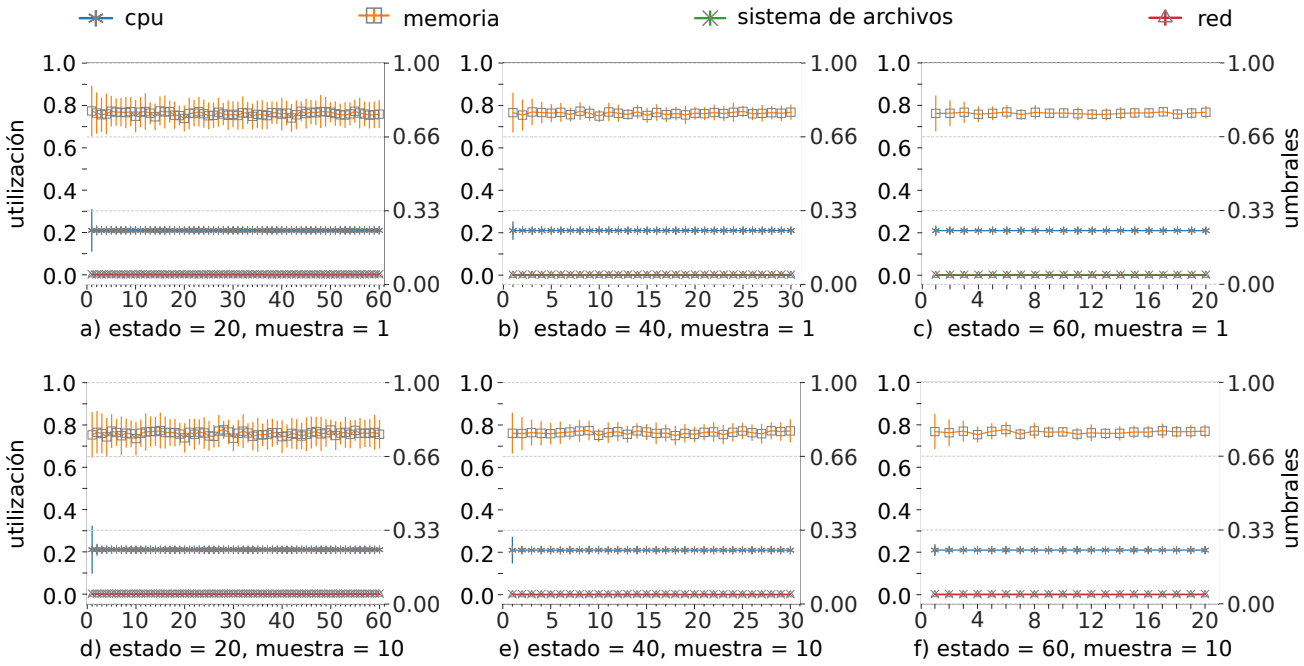


Figura 5.7: Configuraciones de estado y muestra para la solución Kulla con ventana = 1200.

Estadístico	ventana = 1200, muestra = 1											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1860	930	620	1860	930	620	1860	930	620	1860	930	620
min	0.1950	0.1961	0.1983	0.2035	0.2819	0.3124	5.52E-08	5.52E-08	5.52E-08	2.11E-06	2.11E-06	2.11E-06
max	0.7688	0.4519	0.3430	0.8454	0.8191	0.8172	0.0043	0.0043	0.0043	0.0071	0.0039	0.0026
media	0.2100	0.2099	0.2098	0.7418	0.7547	0.7585	0.0023	0.0023	0.0023	1.95E-05	1.84E-05	1.77E-05
mediana	0.2099	0.2100	0.2099	0.7625	0.7637	0.7636	0.0025	0.0025	0.0025	4.21E-06	4.21E-06	4.21E-06
std	0.0135	0.0082	0.0056	0.0668	0.0423	0.0360	0.0013	0.0013	0.0013	0.0002	0.0001	0.0001
escala de riesgo	bajo	bajo	bajo	alto	alto	alto	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	1855	927	613	1541	761	519	1098	578	366	1841	918	610
sigma1_perc	99.73	99.68	98.87	82.85	81.83	83.71	59.03	62.15	59.03	98.98	98.71	98.39
sigma2_n	1857	928	618	1732	901	608	1860	930	620	1850	923	615
sigma2_perc	99.84	99.78	99.68	93.12	96.88	98.06	100.00	100.00	100.00	99.46	99.25	99.19
sigma3_n	1857	928	619	1835	922	616	1860	930	620	1853	926	616
sigma3_perc	99.84	99.78	99.84	98.66	99.14	99.35	100.00	100.00	100.00	99.62	99.57	99.35

Tabla 5.14: Estadísticos descriptivos del experimento con Kulla con ventana = 1200 y muestra = 1.

Estadístico	ventana = 1200, muestra = 10											
	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1860	930	620	1860	930	620	1860	930	620	1860	930	620
min	0.1970	0.1972	0.1986	0.2307	0.2761	0.3286	5.52E-08	5.52E-08	5.52E-08	2.11E-06	2.11E-06	2.11E-06
max	0.8409	0.5591	0.3642	0.8475	0.8403	0.8458	0.0043	0.0043	0.0043	0.0192	0.0048	0.0029
media	0.2101	0.2100	0.2099	0.7377	0.7524	0.7578	0.0023	0.0023	0.0023	2.64E-05	1.97E-05	1.83E-05
mediana	0.2099	0.2099	0.2099	0.7603	0.7638	0.7655	0.0025	0.0024	0.0025	4.22E-06	4.22E-06	4.22E-06
std	0.0153	0.0117	0.0064	0.0704	0.0486	0.0390	0.0013	0.0012	0.0013	0.0005	0.0002	0.0001
escala de riesgo	bajo	bajo	bajo	alto	alto	alto	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	1856	927	615	1392	737	507	1098	625	366	1852	920	611
sigma1_perc	99.78	99.68	99.19	74.84	79.25	81.77	59.03	67.20	59.03	99.57	98.92	98.55
sigma2_n	1856	928	619	1776	895	606	1860	930	620	1856	924	615
sigma2_perc	99.78	99.78	99.84	95.48	96.24	97.74	100.00	100.00	100.00	99.78	99.35	99.19
sigma3_n	1857	928	619	1844	919	616	1860	930	620	1857	926	617
sigma3_perc	99.84	99.78	99.84	99.14	98.82	99.35	100.00	100.00	100.00	99.84	99.57	99.52

Tabla 5.15: Estadísticos descriptivos del experimento con Kulla con ventana = 1200 y muestra = 10.

5.6.2 Incremento en cantidad de soluciones/contenedores

El propósito de este experimento es calcular el tiempo que le toma al prototipo realizar la tarea de supervisión. Para ello se calculan los tiempos de servicio de cada etapa y el tiempo de respuesta que le toma al sistema completo obtener el estado de la(s) soluciones. En este experimento se incrementa la cantidad de soluciones y de contenedores considerando 1 y 10 en ambos casos como se muestra en la Tabla 5.16. La infraestructura utilizada se muestra en la Tabla 5.1. Para construir las soluciones de cada configuración se utilizó el algoritmo IDA dentro un contenedor, replicándolo según la cantidad necesaria de contenedores. El contenedor de IDA realiza de forma secuencial la Dispersión y Recuperación de un archivo de 520MB, con los parámetros $n = 8$, $m = 4$. Estos parámetros de IDA aseguran un mayor uso de los recursos que es lo buscado en este caso. La Dispersión consiste en partir un archivo en n partes, mientras que la Recuperación toma m bloques para recuperar el archivo original.

Configuración	# soluciones	# contenedores
sol1c1	1	1
sol1c10	1	10
sol10c1	10	1
sol10c10	10	10

Tabla 5.16: Configuraciones del experimento variando soluciones y contenedores.

En la Figura 5.8 muestra en el eje horizontal las configuraciones, mientras en el eje vertical se muestra el tiempo de servicio de las etapas del método, donde cada etapa se muestra con un estilo diferente. El tiempo de servicio es calculado de acuerdo a la ecuación 5.1. Se puede observar que la etapa que toma más tiempo en realizarse es la de diagnóstico, lo cual es el resultado del procesamiento del bloque de datos requerido para obtener el estado de las soluciones. El comportamiento que siguen considerando todas las configuraciones es exponencial.

$$ST\_S = \sum_{i=1}^{m} ST_i \tag{5.1}$$

donde:

ST_S = Es el tiempo de servicio de una etapa.

ST_i = Es el tiempo de servicio de la muestra actual.

m = Es la cantidad de muestras realizadas por el monitoreo en un período de tiempo. Donde el período incluye muestras desde un tiempo de inicio hasta un tiempo de fin. Aplica para monitoreo e indexado, en el caso de diagnóstico y representación este valor es 1, ya que no se repiten.

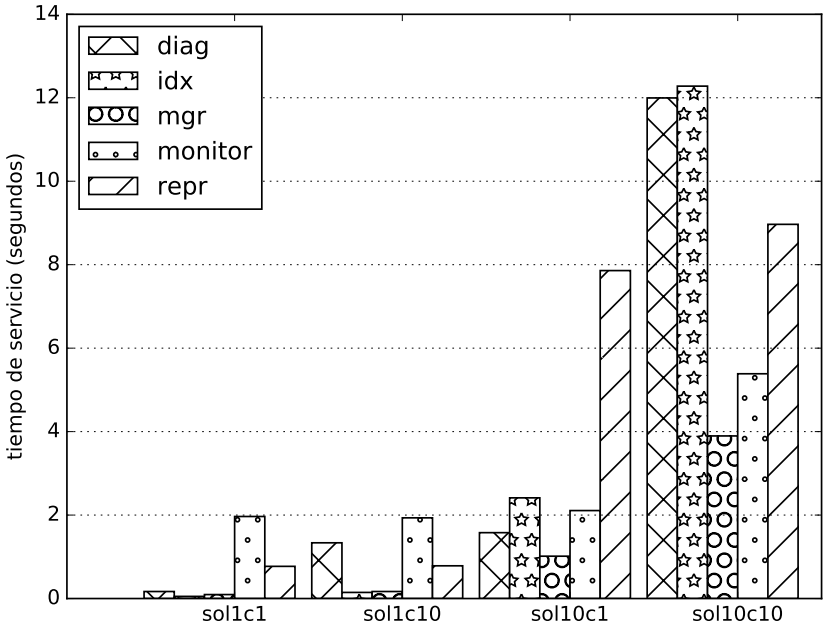


Figura 5.8: Tiempos de servicio de cada etapa por cada configuración.

La Figura 5.9 muestra las configuraciones en el eje horizontal. En el eje vertical muestra los tiempos de respuesta (con base en la ecuación 5.2) desde que se entrega el archivo YAML hasta que se obtiene la respuesta del estado de la solución. Esto aplica para las configuraciones de una solución (sol1c1 y sol1c10). Para las configuraciones de más de una solución (sol10c1 y sol10c10) este proceso se realiza de la siguiente manera. a) Los archivos YAML se agregan de manera secuencial; b) El monitoreo y el indexado de métricas se realiza de manera concurrente. Esto es, todas las soluciones son monitorizadas a la par mediante trabajadores, tomando como base las soluciones; c) Este proceso es realizado repetidas veces con un intervalo de tiempo de 10 segundos; d) El diagnóstico y la representación son realizados de manera secuencial, en ese orden para cada solución. Entonces, el tiempo respuesta para configuraciones de varias soluciones se muestra en la ecuación 5.3. El comportamiento que se sigue es exponencial, esto considerando como base la cantidad de soluciones agregadas, por lo que el incremento de soluciones tiene mayor influencia que el de contenedores.

$$RT\_S = \sum_{i=1}^{S} ST\_S_i + TT_i \tag{5.2}$$

donde:

RT_S = Es el tiempo de respuesta de una solución.

ST_S_i = Es el tiempo de servicio de una etapa de acuerdo a la ecuación 5.8.

TT_i = Es el tiempo de transporte de una etapa, incluyendo el tiempo de carga y descarga.

S = Las etapas del método.

$$RT\_M = \frac{\sum_{i=1}^{M} RT\_S_i}{count(M)} \tag{5.3}$$

donde:

RT_M = Es el tiempo de respuesta cuando hay múltiples soluciones.

M = Son las múltiples soluciones aplicables a la configuración.

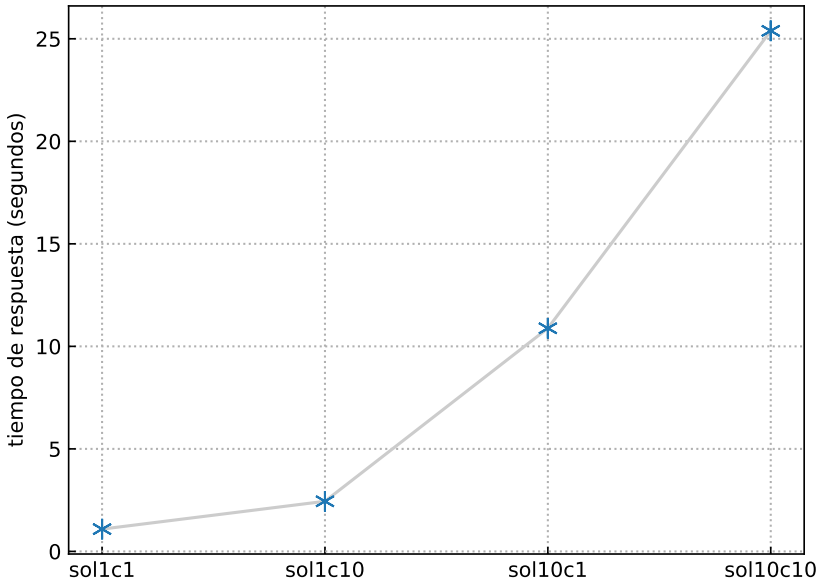


Figura 5.9: Tiempos de respuesta por cada configuración.

5.6.3 Incremento en carga de trabajo

En este experimento se emplearon dos soluciones: IDA y Kulla. Para cada solución se evaluaron diferentes tamaños de ventana, 10, 20 y 30 minutos. Posteriormente se dividieron en bloques de tiempo de 1, 2 y 3 minutos respectivamente. Esto se realizó para mostrar el cambio de estado a

través de cada bloque. Solo se muestra el tiempo de ventana más representativo de cada solución, 30 minutos el caso de IDA y 10 minutos para Kulla. Se utiliza un intervalo de sensado de 10 segundos para obtener cada muestra de las métricas. Las métricas evaluadas corresponden a la utilización del CPU, Sistema de archivos, Memoria y Red, las cuales fueron definidas en el Capítulo 4. La Tabla 5.17 muestra la infraestructura utilizada. Esta corresponde a un computador portátil.

Hostname	Cores	RAM	SSD	OS
localhost	8	8	150 GB	Ubuntu 18

Tabla 5.17: Infraestructura utilizada en este experimento.

5.6.3.1. Experimento con IDA

En este experimento se utilizó un contenedor con una implementación del algoritmo de dispersión IDA [46]. Para IDA se cuenta con dos funciones *Dis* para dispersar un archivo en m bloques y *Rec* para recuperar los datos de un archivo a partir de n de esos bloques. El proceso que se siguió para aplicar carga fue el siguiente. Se realizó la dispersión y recuperación de manera repetitiva utilizando archivos de 520 MB. Este ciclo comienza desde un trabajador y va incrementando de uno en uno hasta llegar a la máxima cantidad de núcleos del equipo, en este caso 8 núcleos. Cada trabajador procesa un archivo diferente de 520 MB. Con esto se asegura el incremento progresivo de la carga de trabajo.

Posteriormente se evaluaron tamaños de ventana de 10 y 20 minutos. Sin embargo, estos no se agregaron ya que los resultados mostraban que no era tiempo suficiente para que se terminara de ejecutar la carga de trabajo con el máximo de núcleos. La mayor carga se logra con un tiempo de ventana de 30 minutos. Los resultados de este experimento se dividen en dos partes en la Figura 5.10.

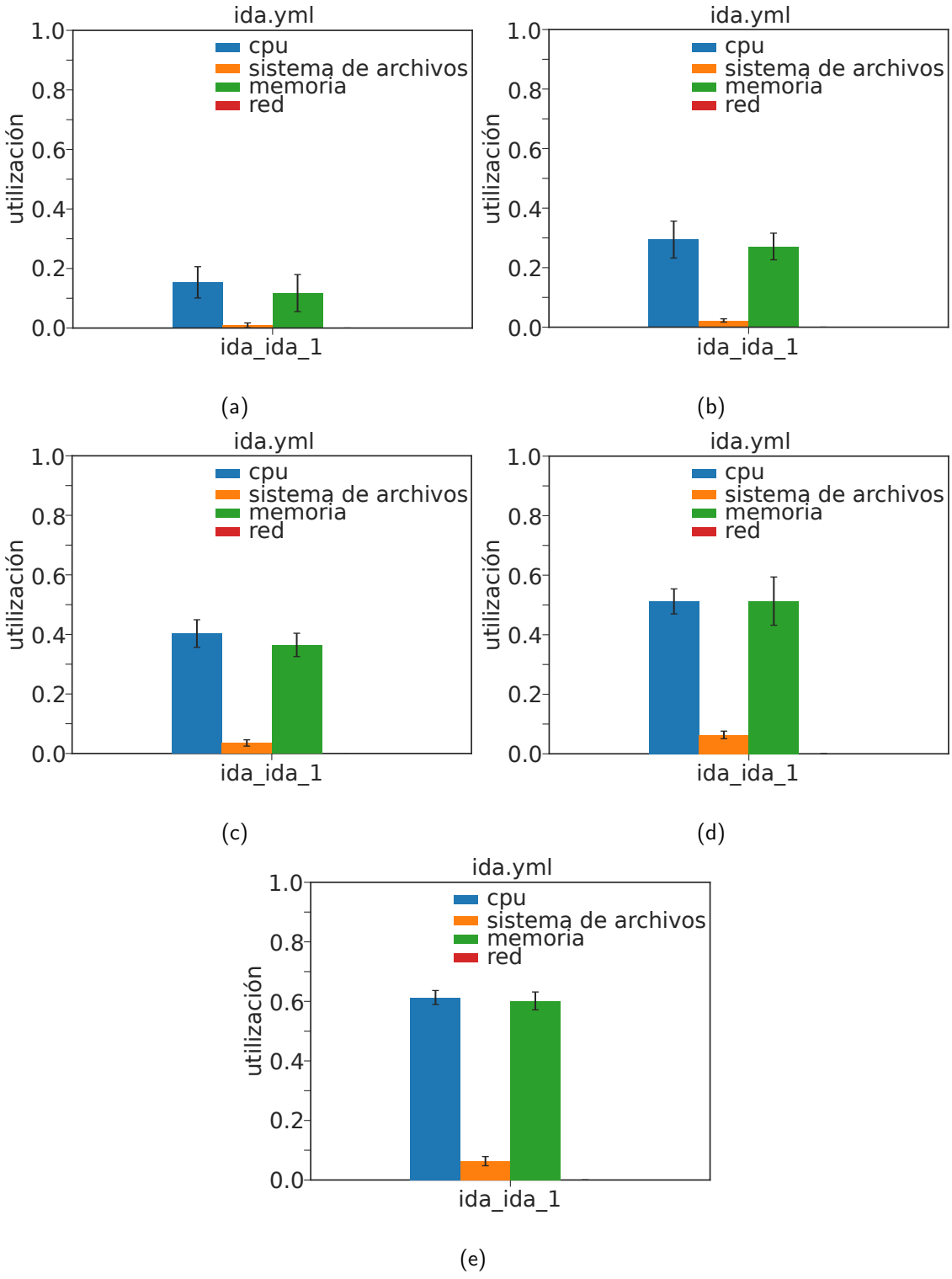


Figura 5.10: Métricas de utilización para el experimento con IDA (Parte 1).

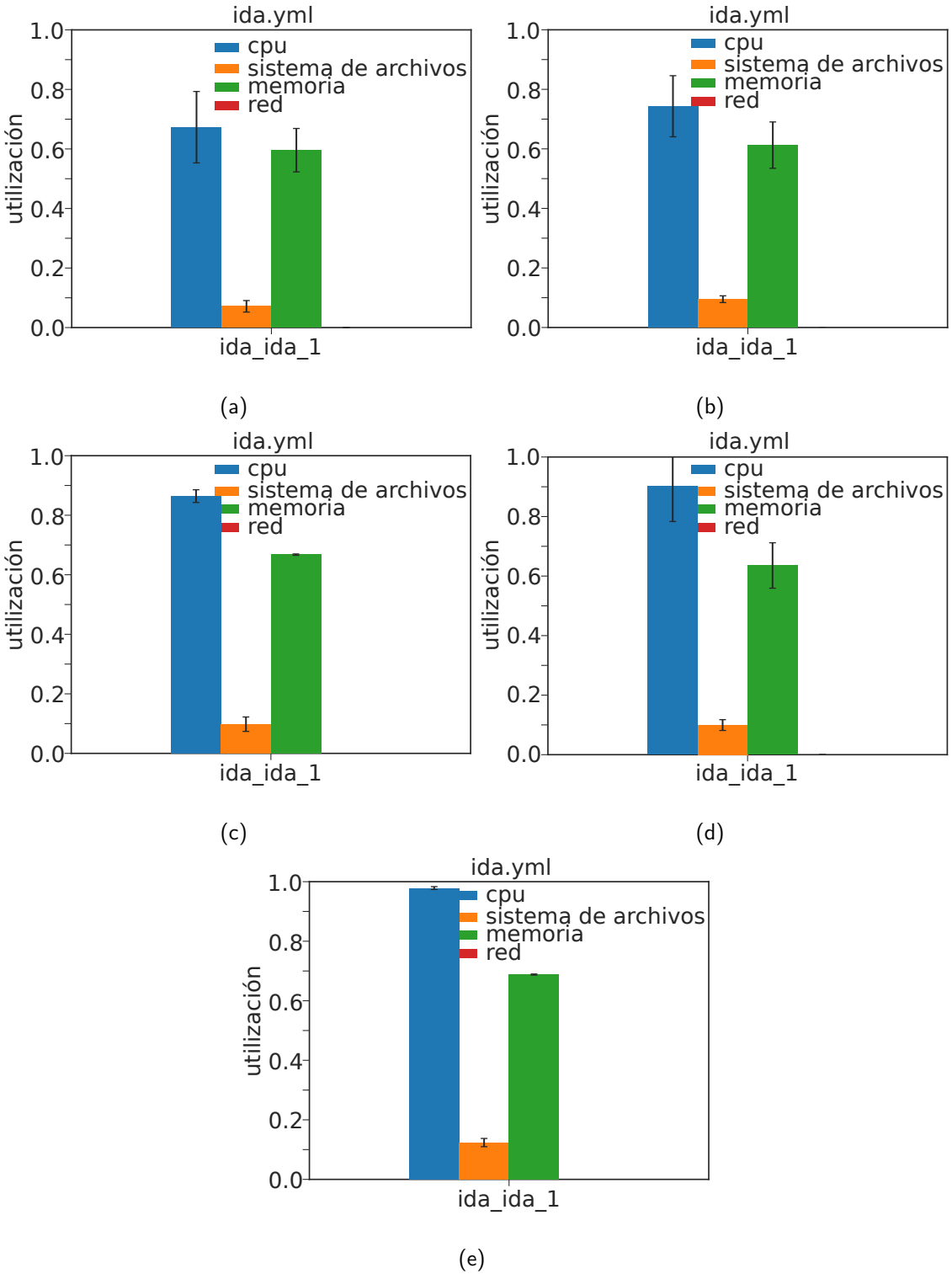


Figura 5.11: Métricas de utilización para el experimento con IDA (Parte 2).

La Figura 5.10 (Parte 1) muestran los primeros 5 bloques y la Figura 5.11 (Parte 2) muestra los 5 bloques restantes. Cada bloque corresponde a 3 minutos de los 30 minutos totales en la ventana, con un total de 10 bloques. Lo que se puede apreciar, considerando la Parte 1 y la 2, es el incremento en los valores de utilización del CPU que van desde un núcleo aproximadamente (en el inciso *a* de la Parte 1) hasta un total de 8 núcleos (inciso *e* de la Parte 2). Otra métrica que se incrementa es la memoria, la cual comienza en 10% de utilización en el primer bloque y va incrementando durante los primeros 5 bloques hasta 60% en el bloque 5 como se muestra en la Tabla 5.20. A partir del bloque 6 al 10 esta métrica se estabiliza entre el 60% y 68%. Al parecer en esta solución no se hace gran uso del sistema de archivos ya que este valor apenas subió 11% pasando de 1% en el bloque 1 al 12% en el bloque 10. Para el caso de la red es una métrica de la cual no se hace uso ya que tiene valores pequeños que pueden ser despreciables e incluso tiene valores nulos como los bloques 8, 9 y 10, como se muestra en la Tabla 5.21.

CPU						
Parte	Bloque	Elemento	media	std	min	max
1	1	a	0.1535	0.0525	0.1153	0.2499
	2	b	0.2943	0.0620	0.2374	0.3749
	3	c	0.4030	0.0462	0.3734	0.4994
	4	d	0.5119	0.0419	0.4592	0.6121
	5	e	0.6130	0.0233	0.5504	0.6248
2	6	a	0.6731	0.1199	0.3230	0.7500
	7	b	0.7431	0.1026	0.3842	0.8728
	8	c	0.8642	0.0213	0.8019	0.8745
	9	d	0.9020	0.1188	0.4840	0.9825
	10	e	0.9785	0.0047	0.9619	0.9821

Tabla 5.18: Estadísticos descriptivos del experimento con IDA en CPU.

Sistema de archivos						
Parte	Bloque	Elemento	media	std	min	max
1	1	a	0.0091	0.0080	0.0000	0.0188
	2	b	0.0226	0.0055	0.0172	0.0299
	3	c	0.0353	0.0105	0.0213	0.0457
	4	d	0.0635	0.0127	0.0459	0.0725
	5	e	0.0632	0.0153	0.0425	0.0771
2	6	a	0.0714	0.0193	0.0464	0.0918
	7	b	0.0949	0.0113	0.0801	0.1066
	8	c	0.0980	0.0242	0.0681	0.1257
	9	d	0.0994	0.0181	0.0751	0.1165
	10	e	0.1236	0.0138	0.1079	0.1408

Tabla 5.19: Estadísticos descriptivos del experimento con IDA en Sistema de archivos.

Memoria

Parte	Bloque	Elemento	media	std	min	max
	1	a	0.1174	0.0622	0.0056	0.2034
	2	b	0.2711	0.0447	0.1733	0
1	3	c	0.3647	0.0391	0.2530	0.4099
	4	d	0.5130	0.0809	0.3663	0.5982
	5	e	0.6015	0.0300	0.5024	0.6196
	6	a	0.5958	0.0729	0.4410	0.6506
	7	b	0.6125	0.0778	0.4337	0.6620
2	8	c	0.6676	0.0023	0.6623	0.6709
	9	d	0.6356	0.0763	0.4474	0.6836
	10	e	0.6880	0.0018	0.6849	0.6914

Tabla 5.20: Estadísticos descriptivos del experimento con IDA en Memoria.

Red

Parte	Bloque	Elemento	media	std	min	max
	1	a	9.23E-06	1.95E-05	0	8.32E-05
	2	b	2.27E-06	1.42E-06	0	4.43E-06
1	3	c	2.19E-06	1.60E-06	0	4.97E-06
	4	d	1.99E-06	1.37E-06	0	2.93E-06
	5	e	1.95E-06	1.42E-06	0	2.93E-06
	6	a	6.85E-07	1.15E-06	0	2.93E-06
	7	b	7.42E-08	2.16E-07	0	6.68E-07
2	8	c	0	0	0	0
	9	d	0	0	0	0
	10	e	0	0	0	0

Tabla 5.21: Estadísticos descriptivos del experimento con IDA en Red.

5.6.3.2. Experimento con Kulla

En este experimento se utilizó un contenedor que ejecuta el patrón Maestro/trabajador de Kulla. Esto es, teniendo un conjunto de archivos y una cantidad de trabajadores, se reparten los archivos entre los trabajadores de modo que cada uno realiza el procesamiento de los archivos que le corresponden. En este patrón se implementa un *pipeline* aplicando los filtros SHA256, LZ4 y AES. Como parámetros configurables se tiene el número de trabajadores que se desea usar en el patrón y el volumen que se utiliza como fuente desde la cual se obtienen los archivos a procesar. El proceso que se siguió para aplicar carga de trabajo fue el siguiente. Se utilizó una fuente de 8 archivos de 520MB, así como una cantidad incremental de trabajadores que va desde 1 hasta 8 trabajadores. Posteriormente se ejecutó el *pipeline* de Kulla. En cada caso se utilizó la misma cantidad de archivos, lo que varió fue la cantidad de trabajadores. Esto implica un incremento en la cantidad de núcleos utilizados.

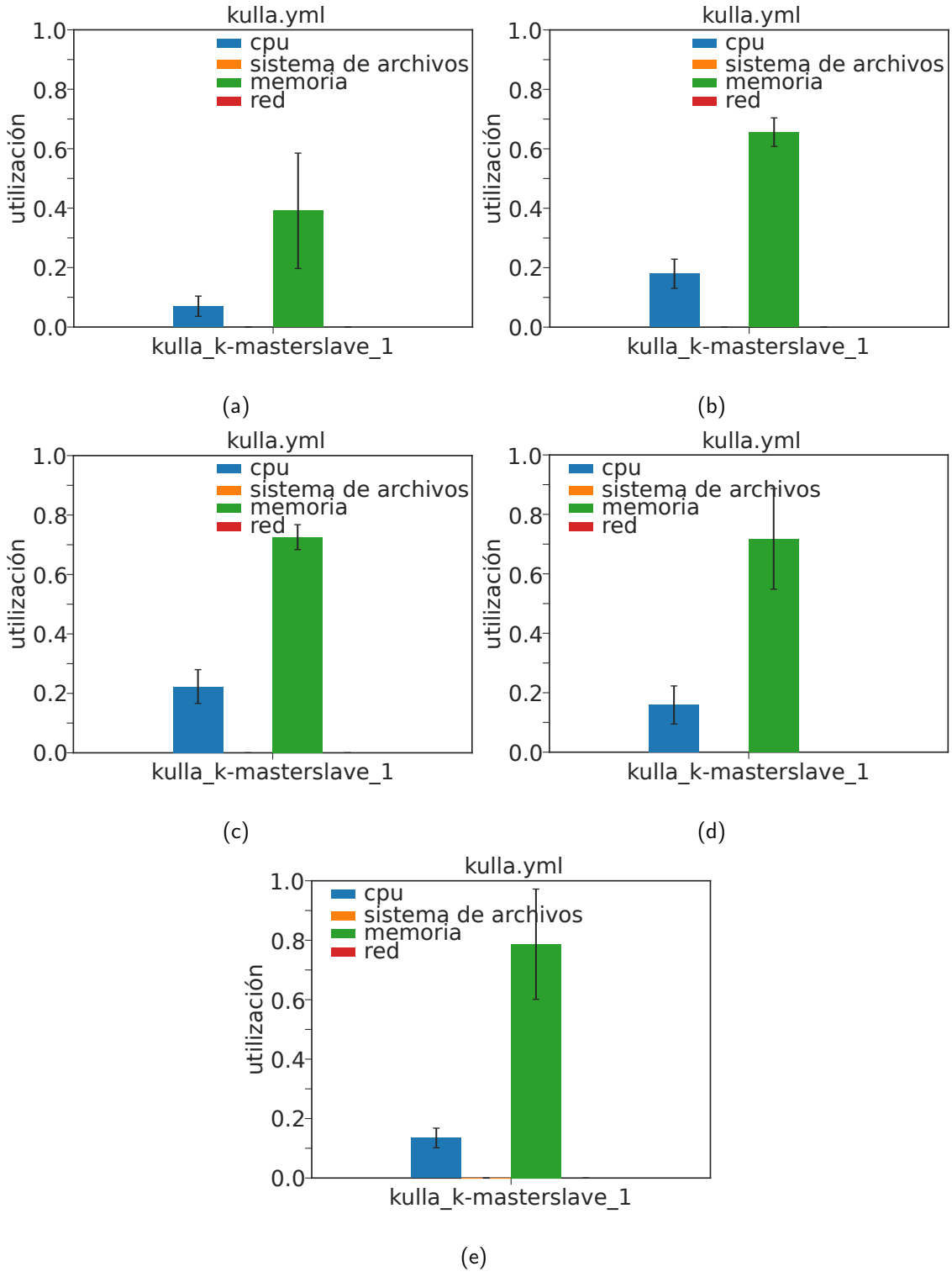


Figura 5.12: Métricas de utilización para el experimento con Kulla (Parte 1).

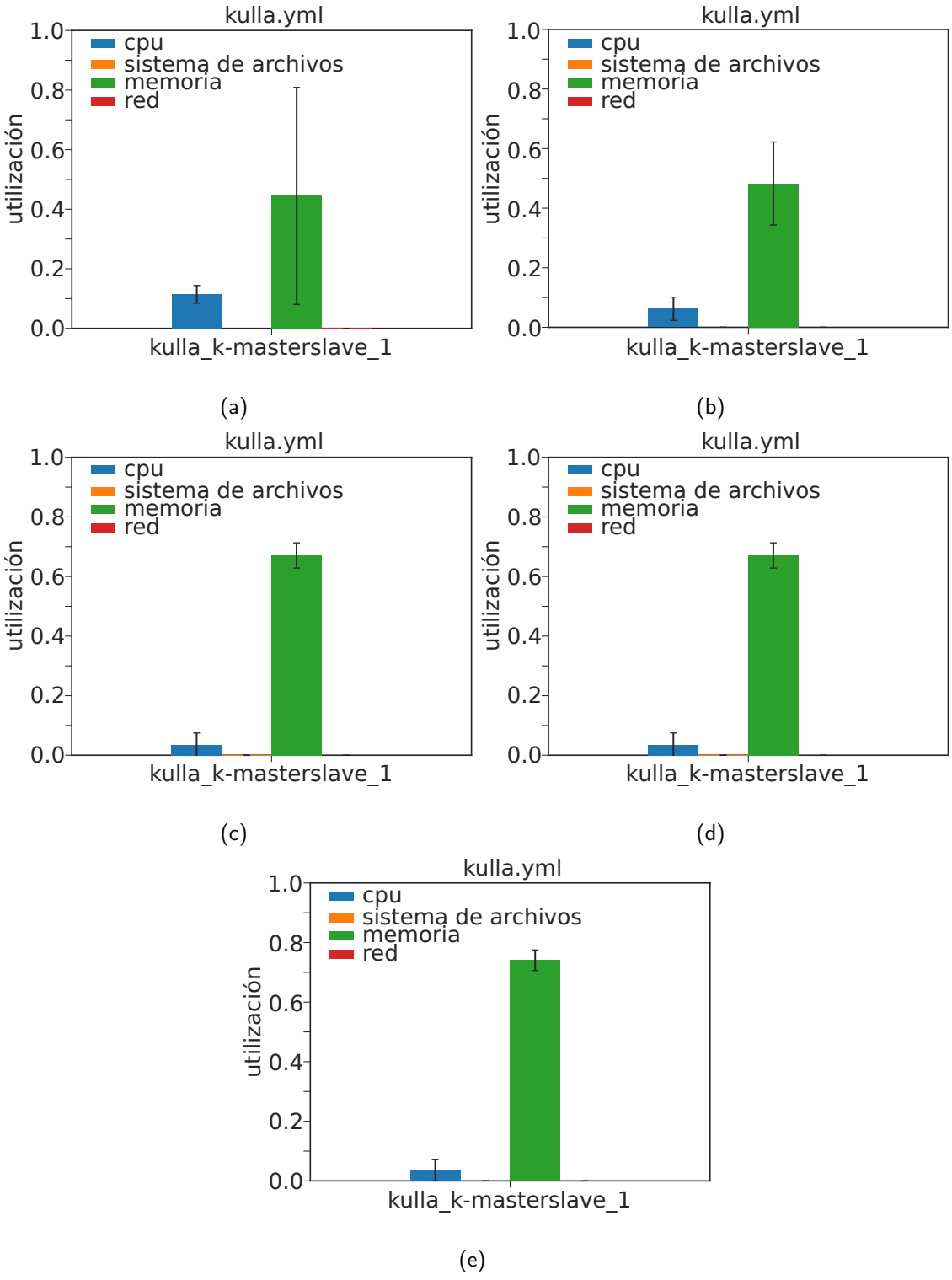


Figura 5.13: Métricas de utilización para el experimento con Kulla (Parte 2).

En este experimento se evaluaron los tamaños de ventana de 10, 20 y 30 minutos. Sin embargo, en los casos de 20 y 30 minutos, se empleó más tiempo del requerido para observar el comportamiento deseado del incremento en la carga de trabajo, por lo que estos experimentos no se muestran. En el caso del tiempo de ventana de 10 minutos, los bloques son de 1 minuto y se tienen dos picos de mayor utilización de la memoria, como se describe a continuación.

La Figura 5.12 (Parte 1) muestra los primeros cinco bloques del experimento, siendo el inciso *e* el que tiene la mayor utilización de memoria, pero tiene un alto valor en la desviación estándar. La Figura 5.13 (Parte 2) tiene los bloques del 6 al 10, en el bloque 6 se tiene la mayor desviación estándar de todos los bloques de ambas partes. Esto se puede deber a dos razones: 1) se tiene un gran incremento en la métrica ó 2) se tiene un gran decremento de la misma. De acuerdo a las gráficas de los bloques anteriores, indican que el comportamiento se debe a un decremento ya que se presenta semejanza entre los bloques 6 al 10 con los bloques 1 al 5. Esto quiere decir que el proceso ha vuelto a iniciar con un trabajador. Este comportamiento es la razón por la que no se agregaron los tamaños de ventana de 20 y 30 minutos.

CPU

Parte	Bloque	Elemento	media	std	min	max
	1	a	0.0701	0.0340	0.0293	0.1066
	2	b	0.1796	0.0491	0.0962	0.2427
1	3	c	0.2229	0.0567	0.1568	0.3003
	4	d	0.1587	0.0640	0.1087	0.2490
	5	e	0.1347	0.0329	0.0953	0.1943
	6	a	0.1143	0.0297	0.0844	0.1538
	7	b	0.0621	0.0392	0.0071	0.0946
2	8	c	0.0350	0.0397	0.0062	0.0917
	9	d	0.0377	0.0331	0.0085	0.0846
	10	e	0.0356	0.0354	0.0077	0.0971

Tabla 5.22: Estadísticos descriptivos del experimento con Kulla en CPU.

Sistema de archivos

Parte	Bloque	Elemento	media	std	min	max
	1	a	1.02E-06	0	1.02E-06	1.02E-06
	2	b	1.28E-06	0	1.28E-06	1.28E-06
1	3	c	1.63E-06	0	1.63E-06	1.63E-06
	4	d	2.01E-06	0	2.01E-06	2.01E-06
	5	e	2.30E-06	0	2.30E-06	2.30E-06
	6	a	2.66E-06	5.60E-08	2.61E-06	2.71E-06
	7	b	2.81E-06	1.12E-07	2.71E-06	2.91E-06
2	8	c	2.91E-06	0	2.91E-06	2.91E-06
	9	d	2.94E-06	2.80E-08	2.91E-06	2.97E-06
	10	e	2.98E-06	2.64E-08	2.97E-06	3.02E-06

Tabla 5.23: Estadísticos descriptivos del experimento con Kulla en Sistema de archivos.

Memoria

Parte	Bloque	Elemento	media	std	min	max
	1	a	0.3910	0.1941	0.0637	0.5516
	2	b	0.6560	0.0480	0.5811	0.7069
1	3	c	0.7256	0.0419	0.6552	0.7858
	4	d	0.7176	0.1692	0.4723	0.8516
	5	e	0.7866	0.1854	0.5365	0.9167
	6	a	0.4446	0.3642	0.0854	0.9156
	7	b	0.4828	0.1393	0.2777	0.6079
2	8	c	0.6705	0.0423	0.6226	0.7251
	9	d	0.6685	0.0261	0.6430	0.7091
	10	e	0.7404	0.0345	0.6943	0.7885

Tabla 5.24: Estadísticos descriptivos del experimento con Kulla en Memoria.

Red

Parte	Bloque	Elemento	media	std	min	max
	1	a	2.88E-05	4.37E-05	0	1.11E-04
	2	b	7.38E-07	1.53E-06	0	3.82E-06
1	3	c	1.07E-06	1.66E-06	0	3.22E-06
	4	d	0	0	0	0
	5	e	8.64E-07	8.94E-07	0	2.61E-06
	6	a	9.74E-06	1.93E-05	0	4.85E-05
	7	b	8.21E-07	6.91E-07	0	1.53E-07
2	8	c	2.56E-07	6.26E-07	0	1.53E-07
	9	d	3.81E-07	6.82E-07	0	1.68E-06
	10	e	4.10E-07	6.64E-07	0	1.53E-06

Tabla 5.25: Estadísticos descriptivos del experimento con Kulla en Red.

5.6.4 Ejecución secuencial y concurrente de aplicaciones

Este experimento evalúa las soluciones *Smallapp* e IDA considerando los casos secuencial y concurrente. Para *Smallapp* se usaron tamaños de 1000, 10000, 100000 que corresponden a la cantidad de números aleatorios a ordenar. En IDA se utilizaron archivos de 10, 100 y 1000 MB. El caso secuencial se refiere a realizar la ejecución de la aplicación con la primer opción, al finalizar se ejecuta la siguiente y así sucesivamente. El caso concurrente lanza todas las opciones al mismo tiempo. En ambos casos se realiza sólo una ejecución sin repeticiones considerando las variaciones en tiempo de estado y muestra. En tiempo de estado las variaciones son 20, 40 y 60 segundos. En tiempo de muestra las variaciones son 1 y 10 segundos.

5.6.4.1. Ejecución secuencial

La Figura 5.14 muestra los resultados obtenidos con la solución *Smallapp* en una ejecución secuencial con las configuraciones de muestra y estado correspondientes. Una vez realizadas las tres

configuraciones termina la ejecución. En cada gráfica el eje vertical izquierdo tiene la utilización, el eje vertical derecho muestra las escalas de riesgo y el eje horizontal son los estados. Los incisos de la fila 1 (a,b y c) corresponden a una muestra de 1 segundo. La fila 2 (incisos d, e y f) tienen una muestra de 10 segundos. Por columnas se tiene una variación en el estado de 20, 40 y 60 para las columnas 1, 2 y 3 respectivamente. Esta solución presentó un tiempo de ejecución de 61 segundos con un comportamiento que se mantuvo dentro de una escala de riesgo bajo en CPU, siendo sus estados bajos en cada configuración. Los datos estadísticos para los casos con muestra = 1 están en la Tabla 5.26, para los casos con muestra = 10 se encuentran en la Tabla 5.27.

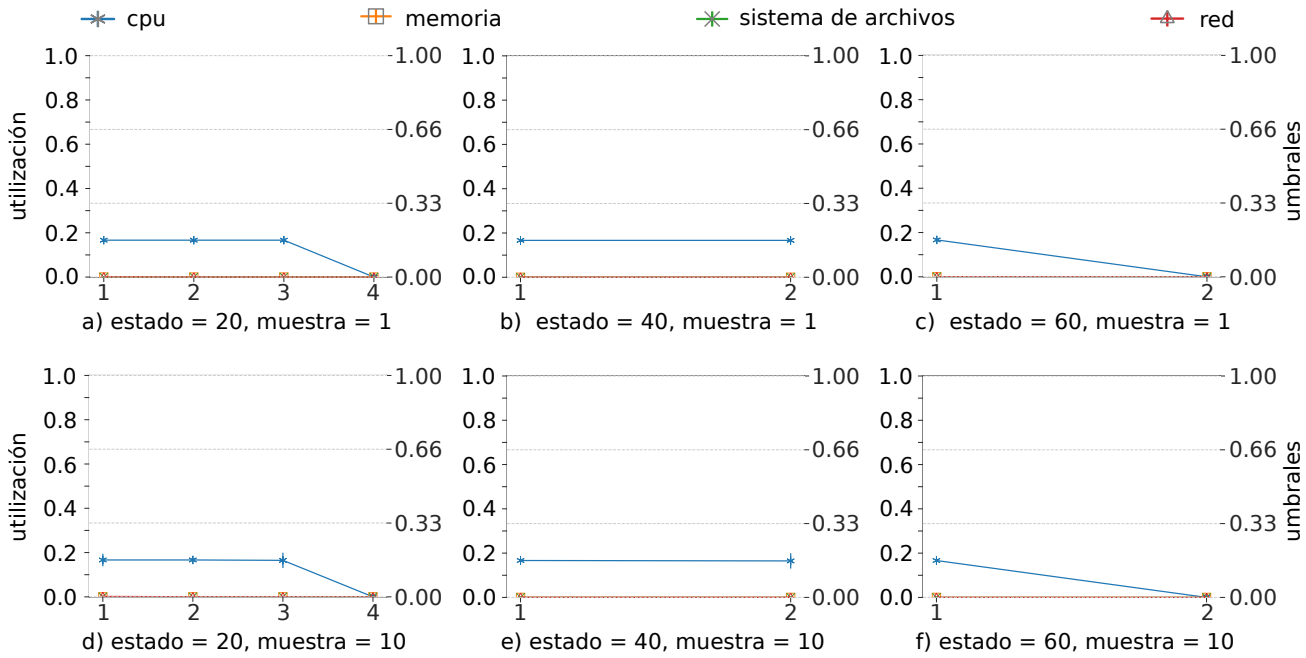


Figura 5.14: Configuraciones de estado y muestra para la solución *Smallapp* con una ejecución secuencial.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	33	20	13	33	20	13	33	20	13	33	20	13
min	0	0.1593	0	0	3.83E-05	0	2.20E-08	2.20E-08	2.20E-08	0	0.0001	0
max	0.1666	0.1666	0.1666	5.05E-05	4.88E-05	4.88E-05	2.48E-05	2.48E-05	2.48E-05	0.0042	0.0021	0.0015
media	0.1497	0.1653	0.1271	4.10E-05	4.48E-05	3.44E-05	4.85E-06	4.99E-06	4.64E-06	0.0009	0.0007	0.0005
mediana	0.1666	0.1666	0.1666	4.85E-05	4.85E-05	4.02E-05	2.97E-06	3.01E-06	2.97E-06	0.0005	0.0004	0.0002
std	0.0482	0.0024	0.0725	1.39E-05	4.85E-06	2.01E-05	6.61E-06	6.97E-06	6.30E-06	0.0011	0.0007	0.0006
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	30	17	10	30	14	10	30	18	12	29	17	10
sigma1_perc	90.91	85.00	76.92	90.91	70.00	76.92	90.91	90.00	92.31	87.88	85.00	76.92
sigma2_n	30	18	13	30	20	13	30	18	12	30	17	13
sigma2_perc	90.91	90.00	100.00	90.91	100.00	100.00	90.91	90.00	92.31	90.91	85.00	100.00
sigma3_n	30	20	13	33	20	13	30	20	12	33	20	13
sigma3_perc	90.91	100.00	100.00	100.00	100.00	100.00	90.91	100.00	92.31	100.00	100.00	100.00

Tabla 5.26: Estadísticos descriptivos de *Smallapp* secuencial con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	33	20	13	33	20	13	33	20	13	33	20	13
min	0	0.0833	0	0	2.43E-05	0	2.20E-08	2.20E-08	2.20E-08	0	7.98E-05	0
max	0.1667	0.1666	0.1666	4.88E-05	4.88E-05	4.88E-05	2.48E-05	2.48E-05	2.48E-05	0.0119	0.0030	0.0018
media	0.1405	0.1564	0.1270	3.89E-05	4.23E-05	3.44E-05	4.85E-06	4.99E-06	4.64E-06	0.0015	0.0008	0.0006
mediana	0.1665	0.1666	0.1666	4.36E-05	4.44E-05	4.02E-05	2.97E-06	3.01E-06	2.97E-06	0.0005	0.0004	0.0003
std	0.0515	0.0252	0.0724	1.44E-05	7.75E-06	2.01E-05	6.61E-06	6.97E-06	6.30E-06	0.0028	0.0010	0.0007
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	28	18	10	28	18	10	30	18	12	29	17	10
sigma1_perc	84.85	90.00	76.92	84.85	90.00	76.92	90.91	90.00	92.31	87.88	85.00	76.92
sigma2_n	30	18	13	30	18	13	30	18	12	31	17	13
sigma2_perc	90.91	90.00	100.00	90.91	90.00	100.00	90.91	90.00	92.31	93.94	85.00	100.00
sigma3_n	33	20	13	33	20	13	30	20	12	31	20	13
sigma3_perc	100.00	100.00	100.00	100.00	100.00	100.00	90.91	100.00	92.31	93.94	100.00	100.00

Tabla 5.27: Estadísticos descriptivos de *Smallapp* secuencial con muestra = 10.

La Figura 5.15 muestra los resultados obtenidos con la solución IDA en una ejecución secuencial. El ejes vertical izquierdo es la utilización, el vertical derecho las escalas de riesgo y el horizontal los

estados. Por columnas de gráficas se varía el estado en 20, 40 y 60, mientras que las filas varían la muestra en 1 y 10. Esta solución presentó un tiempo de ejecución de 461 segundos lo que equivale a 7 minutos con 41 segundos. Durante este tiempo mostró un incremento en el nivel de carga de CPU y Memoria, pasando de bajo a medio. En CPU los primeros dos estados fueron bajos y el resto medios. En Memoria menos de la mitad de tiempo de ejecución mantuvo un estado bajo y el resto medio, lo que indica que el tamaño mayor de números (100000) tomó más tiempo que los tamaños pequeños (1000 y 10000). Los datos estadísticos se muestran en la Tabla 5.28 para los casos con muestra = 1 y en la Tabla 5.29 para los casos con muestra = 10.

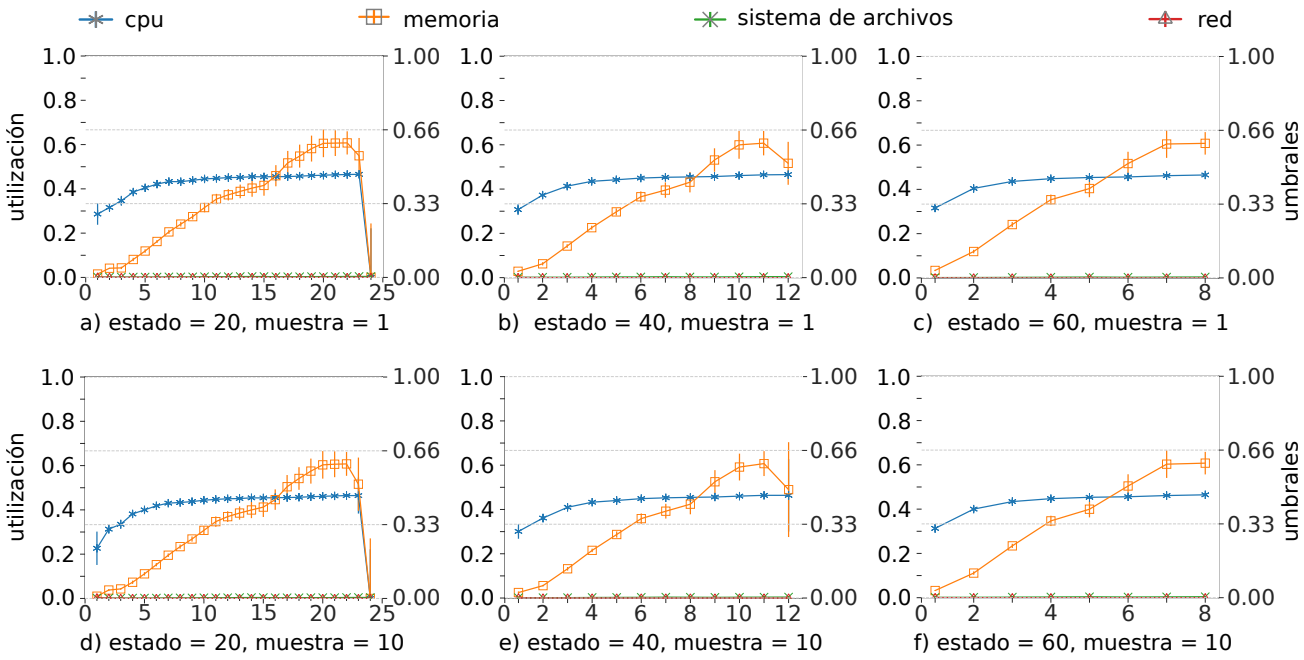


Figura 5.15: Configuraciones de estado y muestra para la solución IDA con una ejecución secuencial.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	191	96	64	191	96	64	191	96	64	191	96	64
min	0	0.2767	0.2929	0	0.0236	0.0303	2.00E-08	2.00E-08	2.00E-08	0	0	0
max	0.4706	0.4706	0.4701	0.6173	0.6170	0.6171	0.0047	0.0047	0.0047	0.0094	0.0042	0.0031
media	0.4172	0.4308	0.4298	0.3280	0.3416	0.3488	0.0029	0.0029	0.0029	0.0003	0.0003	0.0003
mediana	0.4482	0.4505	0.4511	0.3558	0.3662	0.3607	0.0032	0.0032	0.0032	9.36E-05	9.97E-05	9.56E-05
std	0.0841	0.0455	0.0473	0.1952	0.1895	0.1956	0.0012	0.0012	0.0012	0.0009	0.0006	0.0005
escala de riesgo	medio	medio	medio	medio	medio	medio	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	170	83	56	109	53	36	123	64	42	182	89	60
sigma1_perc	89.01	86.46	87.50	57.07	55.21	56.25	64.40	66.67	65.63	95.29	92.71	93.75
sigma2_n	183	88	56	191	96	64	179	90	60	187	92	61
sigma2_perc	95.81	91.67	87.50	100.00	100.00	100.00	93.72	93.75	93.75	97.91	95.83	95.31
sigma3_n	186	93	64	191	96	64	191	96	64	187	93	62
sigma3_perc	97.38	96.88	100.00	100.00	100.00	100.00	100.00	100.00	100.00	97.91	96.88	96.88

Tabla 5.28: Estadísticos descriptivos de IDA secuencial con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	191	96	64	191	96	64	191	96	64	191	96	64
min	0	0	0.2869	0	0	0.0298	2.00E-08	2.00E-08	2.00E-08	0	0	0
max	0.4708	0.4705	0.4700	0.6177	0.6169	0.6166	0.0047	0.0047	0.0047	0.0230	0.0061	0.0036
media	0.4129	0.4237	0.4286	0.3221	0.3303	0.3441	0.0029	0.0029	0.0029	0.0005	0.0003	0.0003
mediana	0.4462	0.4496	0.4508	0.3525	0.3587	0.3575	0.0032	0.0032	0.0032	9.67E-05	9.85E-05	9.64E-05
std	0.0889	0.0655	0.0484	0.1954	0.1930	0.1962	0.0012	0.0012	0.0012	0.0020	0.0008	0.0006
escala de riesgo	medio	medio	medio	medio	medio	medio	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	171	83	56	108	52	36	123	64	42	186	91	60
sigma1_perc	89.53	86.46	87.50	56.54	54.17	56.25	64.40	66.67	65.63	97.38	94.79	93.75
sigma2_n	181	91	59	191	96	64	179	90	60	188	93	61
sigma2_perc	94.76	94.79	92.19	100.00	100.00	100.00	93.72	93.75	93.75	98.43	96.88	95.31
sigma3_n	186	95	64	191	96	64	191	96	64	188	93	62
sigma3_perc	97.38	98.96	100.00	100.00	100.00	100.00	100.00	100.00	100.00	98.43	96.88	96.88

Tabla 5.29: Estadísticos descriptivos de IDA secuencial con muestra = 10.

5.6.4.2. *Ejecución concurrente*

La Figura 5.16 muestra los resultados de la solución *Smallapp* con una ejecución concurrente. El eje vertical izquierdo muestra la utilización y el derecho las escalas de riesgo. El eje horizontal son los estados. Las gráficas están ordenadas por columnas por la variación de estado en 20, 40 y 60 segundos. Las filas se ordenan por la variación de muestra en 1 y 10 segundos. Esta solución presentó un tiempo de ejecución de 68 segundos. Conserva un nivel de carga bajo en todo momento, con estados bajos en cada caso. Los datos estadísticos con muestra = 1 se pueden ver en la Tabla 5.30 y con muestra = 10 en la Tabla 5.31.

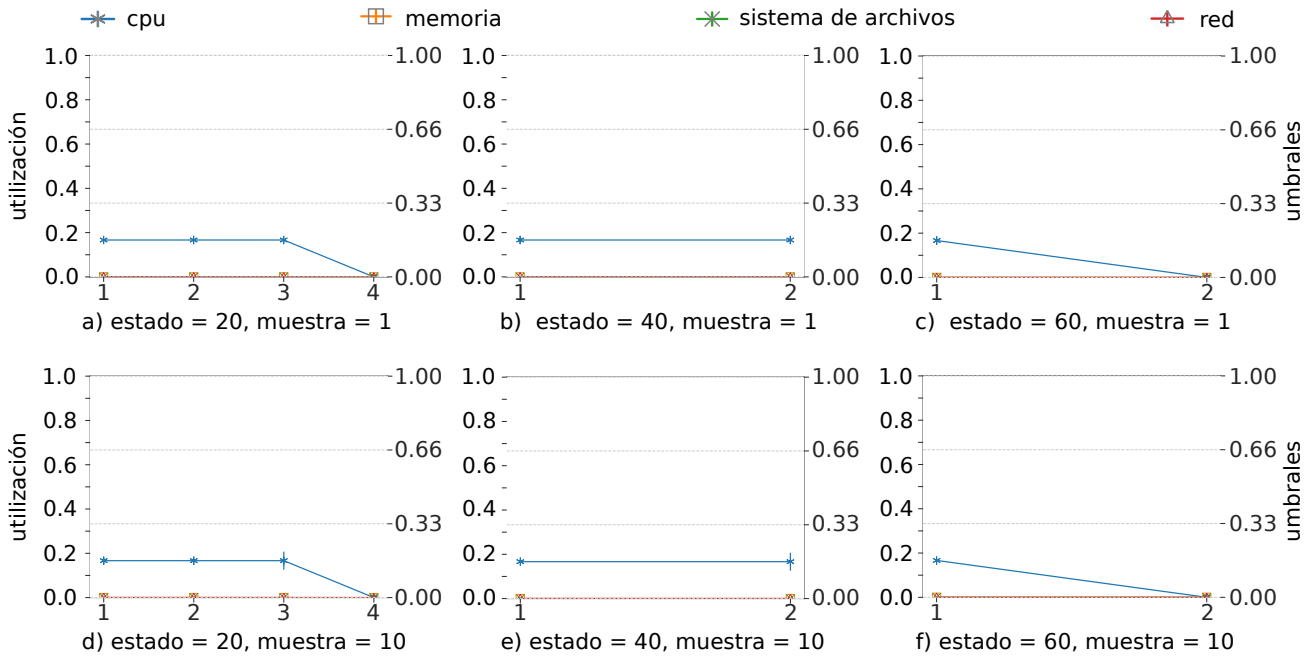


Figura 5.16: Configuraciones de estado y muestra para la solución *Smallapp* con una ejecución concurrente.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	22	14	8	22	14	8	22	14	8	22	14	8
min	0	0.1648	0	0	4.52E-05	0	2.20E-08	2.20E-08	2.20E-08	0	0	0
max	0.1709	0.1688	0.1683	6.79E-05	6.79E-05	6.79E-05	8.15E-06	8.15E-06	8.15E-06	0.0053	0.0024	0.0017
media	0.1591	0.1667	0.1458	4.75E-05	4.98E-05	4.36E-05	5.46E-06	5.69E-06	5.07E-06	0.0009	0.0006	0.0006
mediana	0.1666	0.1666	0.1666	4.55E-05	4.55E-05	4.54E-05	6.33E-06	6.33E-06	5.96E-06	0.0004	0.0003	0.0003
std	0.0356	0.0009	0.0589	1.33E-05	8.29E-06	1.93E-05	2.76E-06	2.65E-06	3.09E-06	0.0013	0.0007	0.0006
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	21	10	7	18	12	6	18	12	6	20	12	6
sigma1_perc	95.45	71.43	87.50	81.82	85.71	75.00	81.82	85.71	75.00	90.91	85.71	75.00
sigma2_n	21	12	7	21	12	7	22	12	8	20	13	8
sigma2_perc	95.45	85.71	87.50	95.45	85.71	87.50	100.00	85.71	100.00	90.91	92.86	100.00
sigma3_n	21	14	8	21	14	8	22	14	8	21	14	8
sigma3_perc	95.45	100.00	100.00	95.45	100.00	100.00	100.00	100.00	100.00	95.45	100.00	100.00

Tabla 5.30: Estadísticos descriptivos de *Smallapp* concurrente con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	22	14	8	22	14	8	22	14	8	22	14	8
min	0	0.0833	0	0	2.26E-05	0	2.20E-08	2.20E-08	2.20E-08	0	0	0
max	0.1862	0.1696	0.1684	7.29E-05	6.79E-05	6.79E-05	8.15E-06	8.15E-06	8.15E-06	0.0130	0.0035	0.0021
media	0.1513	0.1548	0.1458	4.57E-05	4.66E-05	4.36E-05	5.46E-06	5.69E-06	5.07E-06	0.0015	0.0007	0.0006
mediana	0.1666	0.1666	0.1666	4.54E-05	4.54E-05	4.54E-05	6.33E-06	6.33E-06	5.96E-06	0.0003	0.0003	0.0003
std	0.0422	0.0303	0.0589	1.56E-05	1.29E-05	1.93E-05	2.76E-06	2.65E-06	3.09E-06	0.0031	0.0010	0.0007
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	19	12	7	16	10	6	18	12	6	20	12	6
sigma1_perc	86.36	85.71	87.50	72.73	71.43	75.00	81.82	85.71	75.00	90.91	85.71	75.00
sigma2_n	21	12	7	21	14	7	22	12	8	20	13	8
sigma2_perc	95.45	85.71	87.50	95.45	100.00	87.50	100.00	85.71	100.00	90.91	92.86	100.00
sigma3_n	21	14	8	22	14	8	22	14	8	21	14	8
sigma3_perc	95.45	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	95.45	100.00	100.00

Tabla 5.31: Estadísticos descriptivos de *Smallapp* concurrente con muestra = 10.

La Figura 5.17 muestra los resultados obtenidos con IDA en una ejecución concurrente. El eje vertical izquierdo muestra la utilización, el derecho las escalas de riesgo y el horizontal inferior los

estados. Las gráficas se ordenan por columnas por la variación de estado en 20, 40 y 60 segundos, se ordenan las filas por la variación de muestra en 1 y 10 segundos. Con un tiempo de ejecución de 453 segundos equivalente a 7 minutos con 33 segundos. Durante este tiempo mostró un decremento en CPU pasando de alto a medio. En memoria mostró un incremento pasando de bajo a medio. Lo que corresponde en los primeros momentos de ejecución a todos los tamaños de archivo corriendo al mismo tiempo. Cuando comienza a disminuir el CPU es cuando terminan los tamaños pequeños (10 y 100 MB), el resto de la ejecución corresponde al tamaño mayor (1000 MB) por lo que la Memoria se ve incrementada en los últimos momentos de ejecución. Las Tablas 5.32 y 5.33 contienen los datos estadísticos para los casos con muestra = 1 y muestra = 10 respectivamente.

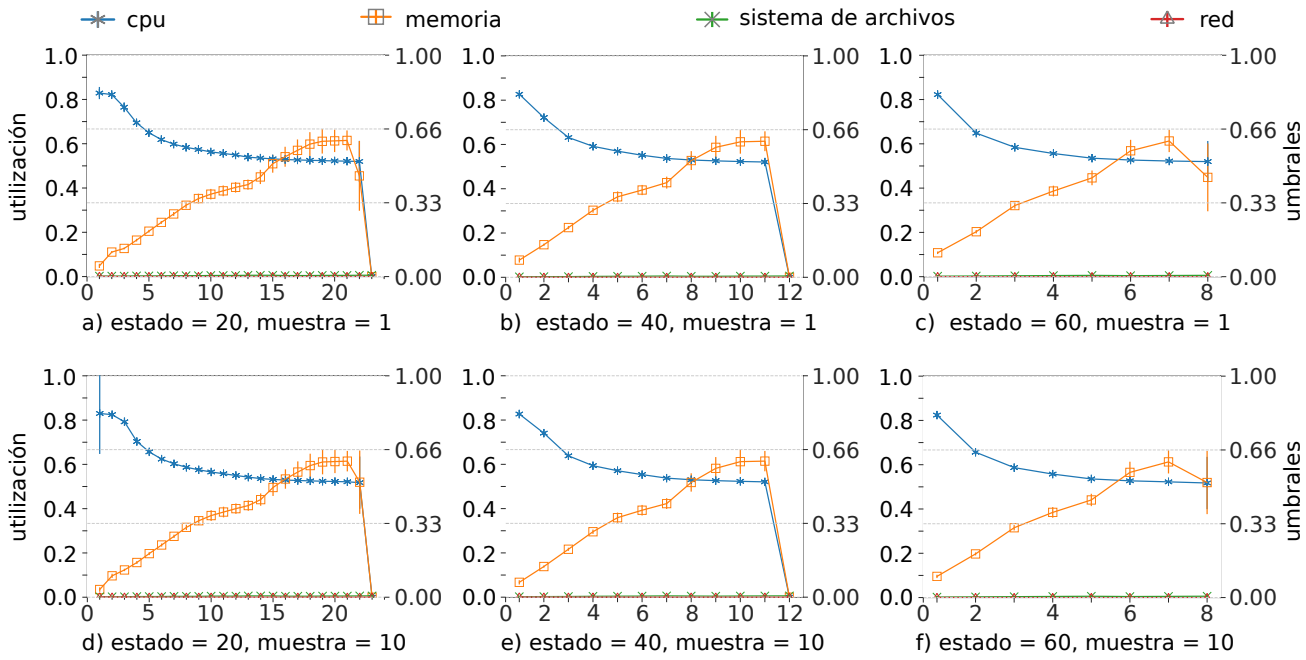


Figura 5.17: Configuraciones de estado y muestra para la solución IDA con una ejecución concurrente.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	177	89	64	177	89	64	177	89	64	177	89	64
min	0	0	0.2578	0	0	0.0976	2.89E-08	2.89E-08	2.89E-08	0	0	0
max	0.8756	0.8485	0.8359	0.6352	0.6351	0.6341	0.0061	0.0061	0.0061	0.0043	0.0023	0.0014
media	0.5897	0.5869	0.5856	0.3713	0.3767	0.3784	0.0042	0.0042	0.0044	0.0002	0.0002	0.0001
mediana	0.5521	0.5512	0.5484	0.3837	0.3849	0.3910	0.0046	0.0046	0.0047	6.02E-05	6.03E-05	5.60E-05
std	0.1080	0.1125	0.1054	0.1737	0.1792	0.1664	0.0014	0.0013	0.0014	0.0004	0.0003	0.0002
escala de riesgo	medio	medio	medio	medio	medio	medio	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	150	75	55	104	52	33	118	65	39	167	82	57
sigma1_perc	84.75	84.27	85.94	58.76	58.43	51.56	66.67	73.03	60.94	94.35	92.13	89.06
sigma2_n	161	80	55	176	88	64	171	86	62	172	87	62
sigma2_perc	90.96	89.89	85.94	99.44	98.88	100.00	96.61	96.63	96.88	97.18	97.75	96.88
sigma3_n	175	88	63	177	89	64	174	88	63	175	87	62
sigma3_perc	98.87	98.88	98.44	100.00	100.00	100.00	98.31	98.88	98.44	98.87	97.75	96.88

Tabla 5.32: Estadísticos descriptivos de IDA concurrente con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	177	89	64	177	89	64	177	89	64	177	89	64
min	0	0	0.2632	0	0	0.0821	2.89E-08	2.89E-08	0.0000	0	0	0
max	0.9272	0.8600	0.8417	0.6353	0.6341	0.6341	0.0061	0.0060	0.0061	0.0087	0.0030	0.0018
media	0.5888	0.5901	0.5834	0.3667	0.3712	0.3769	0.0042	0.0042	0.0044	0.0002	0.0002	0.0001
mediana	0.5526	0.5533	0.5492	0.3802	0.3808	0.3881	0.0046	0.0046	0.0047	5.89E-05	6.15E-05	5.00E-05
std	0.1140	0.1152	0.1135	0.1760	0.1813	0.1695	0.0014	0.0014	0.0014	0.0008	0.0004	0.0003
escala de riesgo	medio	medio	medio	medio	medio	medio	bajo	bajo	bajo	bajo	bajo	bajo
sigma1_n	147	73	54	103	52	32	118	56	39	171	83	59
sigma1_perc	83.05	82.02	84.38	58.19	58.43	50.00	66.67	62.92	60.94	96.61	93.26	92.19
sigma2_n	163	82	56	176	88	64	171	86	62	173	87	62
sigma2_perc	92.09	92.13	87.50	99.44	98.88	100.00	96.61	96.63	96.88	97.74	97.75	96.88
sigma3_n	176	88	64	177	89	64	174	88	63	175	87	62
sigma3_perc	99.44	98.88	100.00	100.00	100.00	100.00	98.31	98.88	98.44	98.87	97.75	96.88

Tabla 5.33: Estadísticos descriptivos de IDA concurrente con muestra = 10.

5.7 Estudio de caso: Gestor de servicios de procesamiento transversal aplicado a datos atmosféricos

En esta sección se describen las pruebas realizadas al prototipo del método utilizando una solución para servicios transversales que gestionan el flujo de datos atmosféricos pasando por diferentes etapas de una estructura de procesamiento.

5.7.1 Gestor de servicios de procesamiento transversal

Este experimento se realizó para determinar el comportamiento del prototipo cuando se procesa un trabajo colaborativo entre diferentes estructuras de procesamiento. Este trabajo colaborativo se denomina servicios de procesamiento transversal (Transversal Processing Service, TPS por sus siglas en Inglés). Un servicio es transversal cuando permite la reutilización de la información generada en sus etapas para que sea reutilizada por otras etapas, ya sean internas o externas. Se distinguen dos tipos de servicios transversales: puntos de acoplamiento (intersecciones abstractas entre tareas) y puntos de extracción (espacios para entregar y extraer datos de las tareas existentes). El Gestor de servicios transversales se encarga de coordinar la integración tanto de puntos de acoplamiento como de intersección y que éstos sean publicados como servicios. El Gestor cuenta con un API mediante la cual se pueden conectar los servicios publicados. Entre las tareas del Gestor se encuentra la extracción de información de las fuentes de datos, indexar la información extraída y gestionar la ejecución de los servicios. Los servicios del Gestor fueron implementados siguiendo el enfoque de microservicios, esto es que cada servicio es implementado de forma independiente manteniendo la comunicación mediante mensajes, en este caso REST. Los servicios se listan a continuación:

- Manager: Servicio en cargado de la gestión, indexación y ejecución del resto de servicios. Proporciona herramientas para el filtrado de datos, transformación y acceso.
- Clustering: Ofrece algoritmos de agrupamiento no supervisado.

- Resume: Ofrece servicios de analítica básica para conjuntos de datos.
- Cleaning Tools: Servicio para la limpieza de datos, ofrece herramientas para la eliminación de datos atípicos, omisión, remplazo de datos (mediante interpolación, media, mediana o moda). Ofrece herramientas para la reestructuración y ordenamiento de datos.
- Graphics: Servicio para la generación de gráficos estadísticos en dos y tres dimensiones (linear, histograma, puntos, clusters, etc.).

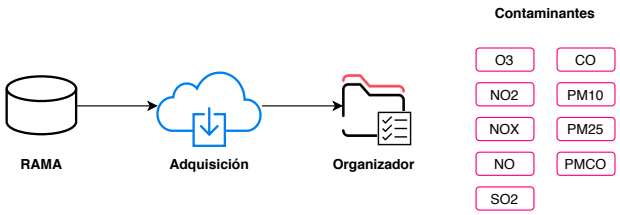


Figura 5.18: Tubería de adquisición RAMA.

Para este caso se utilizó la tubería de procesamiento de RAMA[1], la cual consiste en la adquisición de datos desde la fuente primaria así como la organización de los mismos (Figura 5.18). Sin embargo, estos datos son considerados sucios ya que contienen valores atípicos y tienen un formato no aceptable para el análisis de datos. Para resolver este problema se hizo uso de servicios transversales para construir un flujo de trabajo que realice este procesamiento de datos. Este consta de tres tareas:

- Clean (C): Realiza un proceso de limpieza de los datos, eliminando o reemplazando los datos faltantes y/o atípicos.
- Transform (T): Transforma la estructura de la tabla de datos. Esto es, las columnas pasan a ser registros.
- Group (G): Agrupa un conjunto de registros y los transforma en uno, obteniendo la media, la mediana o la moda. Esto es, los registros de la misma antena y el mismo día (pero diferente tiempo de detección) se agruparon en uno usando la media.

[1]http://www.aire.cdmx.gob.mx/default.php?opc=%27ZaBhnmI=%27

Se aplicó la tubería de procesamiento de tres tareas a cada variable producida por RAMA para después unificarlas en una sola tabla de contaminantes. Estas estructura se unen mediante el Gestor de servicios transversales que en sus etapas finales realizan un análisis de datos sobre la tabla unificada a fin de obtener información útil (Figura 5.19).

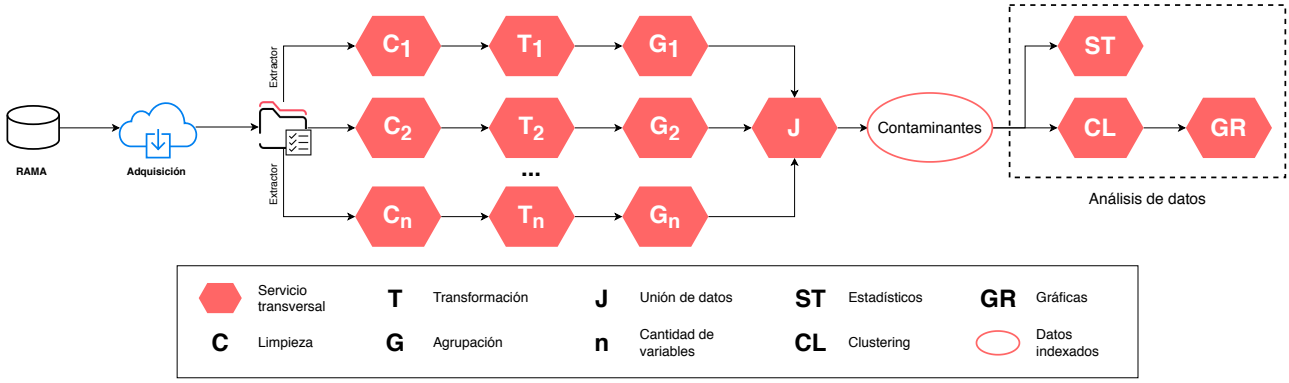


Figura 5.19: Estructura completa de RAMA.

5.7.2 Datos atmosféricos

Se utilizaron datos atmosféricos de la Red Automática de Monitoreo Atmosférico (RAMA) obtenidos del Sistema de Monitoreo Atmosférico (SIMAT) de la Ciudad de México. RAMA contiene bases de datos por año con información sobre la concentración de contaminantes que se han registrado cada hora desde 1986. Estos contaminantes se puede descargar en archivos .xls comprimidos desde la página oficial, contienen los contaminantes mostrados en la Tabla 5.34. Los datos utilizados comprenden un período de cuatro años entre 2015 y 2019.

5.7.3 Resultados del diagnóstico

Este apartado corresponde a los resultados obtenidos con el Gestor de servicios transversales, el cual realizó las tareas mostradas en la Figura 5.19, esto ejecutándose sobre el equipo con la etiqueta compute11 de la Tabla 5.1. Estos resultados toman como base un núcleo, ya que se observó que el gestor tiene un desempeño que no rebasa esa cantidad de núcleos. La Figura 5.20 muestran las

Abreviatura	Contaminante (Símbolo)	Unidad de medición
O3	Ozono (O_3)	ppb
NO2	Dióxido de nitrógeno (NO_2)	ppb
NOX	Óxidos de nitrógeno (NO_X)	ppb
NO	Monóxido de nitrógeno (NO)	ppb
SO2	Dióxido de azufre (SO_2)	ppb
CO	Monóxido de carbono (CO)	ppm
PM10	Partículas menores a 10 micrómetros (PM_{10})	$\mu g/m^3$
PM25	Partículas menores a 2.5 micrómetros ($PM_{2,5}$)	$\mu g/m^3$
PMCO	Partículas fracción gruesa o "coarse" ($PM_{10-2,5}$)	$\mu g/m^3$

ppb = partes por billón, ppm = partes por millón, $\mu g/m^3$ = microgramos/metro cúbico

Tabla 5.34: Contaminantes medidos en RAMA.

gráficas por cada contenedor del Gestor, los descritos en la sección 5.7.1, con un tiempo de muestra = 1. Los contenedores se encuentran por filas de gráficas. Por columnas de gráficas se varía el tiempo de estado en 20, 40 y 60 segundos. Cada gráfica tiene en su eje vertical izquierdo la utilización en un rango de 0 y 1. El eje vertical derecho son las escalas de riesgo, 1 dividido en tres partes iguales que de menor a mayor corresponden a bajo, medio y alto. El eje horizontal son los estados. Los contenedores *DB_data* y *cleaning_tools* tienen un nivel de carga que pasa de bajo (en los primeros momentos) a medio (en el resto de ejecución). Los contenedores *clustering*, *graphics* y *summary* presentan un nivel bajo durante toda la ejecución excepto en el inicio. Claramente el contenedor que involucra una mayor carga es el *Manager* incrementando de bajo, medio y alto en Red, mientras que para CPU comienza en medio e incrementa a alto donde se mantiene más tiempo de ejecución.

La Figura 5.21 muestra los resultados de cada contenedor del Gestor con un tiempo de muestra = 10. Esto se refiere a que se consideran menos muestras ya que se toma una cada 10 segundos. Esto a fin de comprobar si los resultados son semejantes a los obtenidos con la muestra = 1. Se puede observar que los niveles de las métricas se conservan, sin embargo con la muestra = 10 se tienen menos oscilaciones debido a la reducción de valores. Al igual que el caso anterior se tiene un nivel de carga entre bajo y medio para los contenedores *DB_data* y *cleaning_tools*. Un nivel bajo en los contenedores *clustering*, *graphics* y *summary*. El *Manager* con niveles de red bajo, medio y

alto, así como medio y alto en CPU. Las Tablas 5.35 a 5.40 muestran los datos estadísticos de los contenedores en la Figura 5.20 (y continuación). Por otra parte, las Tablas 5.41 a 5.46 muestran los datos estadísticos de los contenedores en la Figura 5.21 (y continuación).

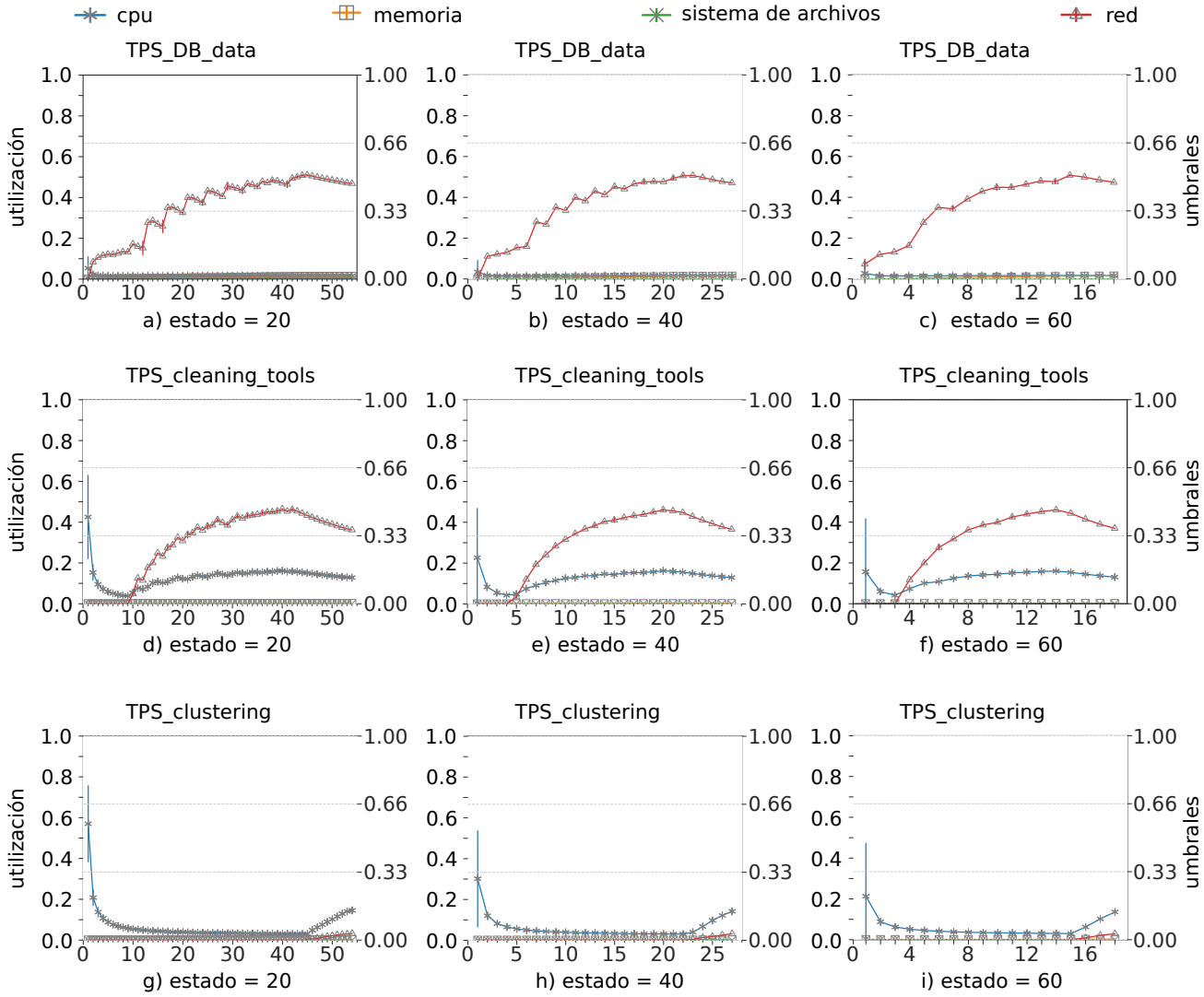


Figura 5.20: Resultados solución TPS con muestra = 1 y estado = 20, 40 y 60.

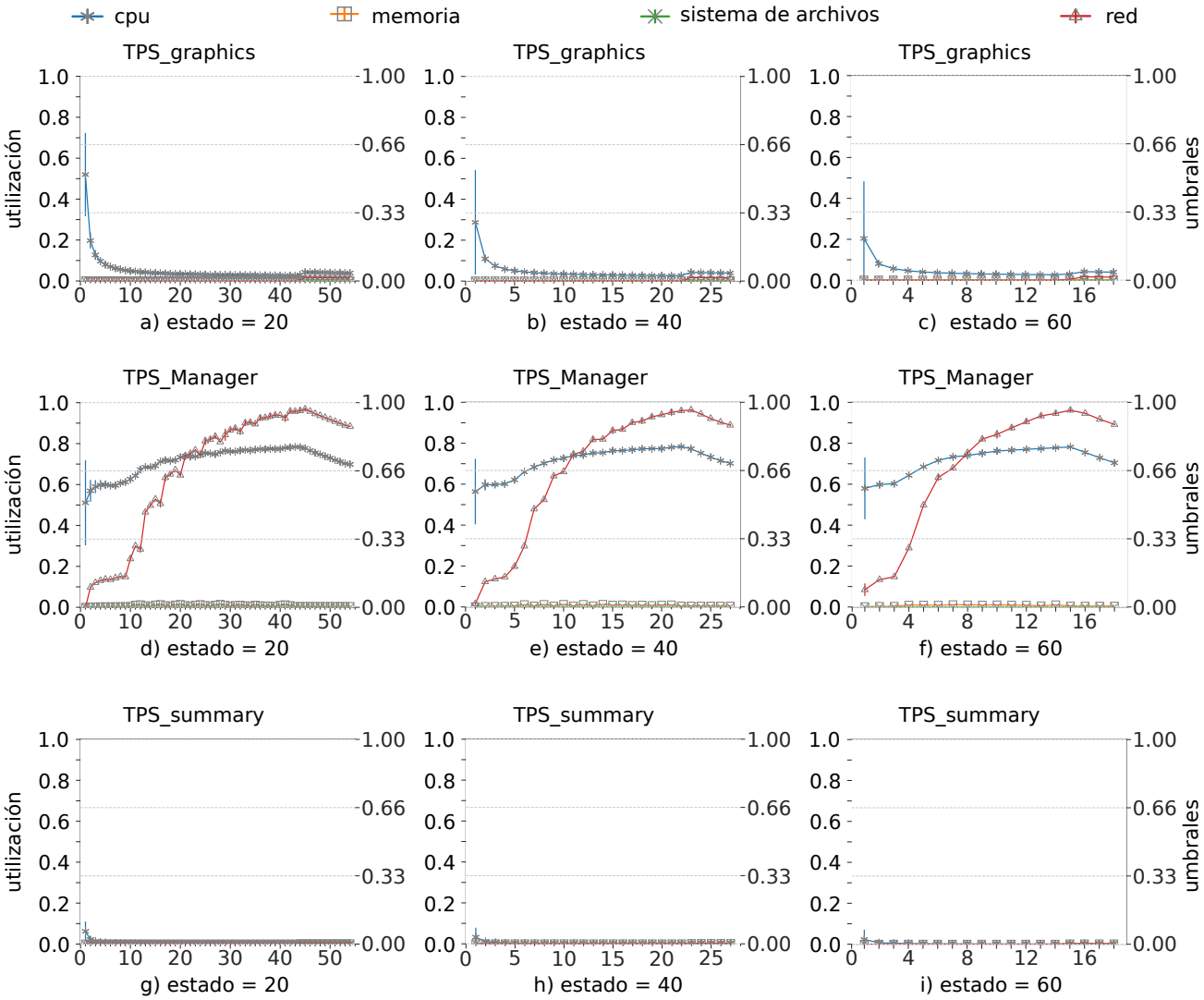


Figura 5.20: Resultados solución TPS con muestra = 1 y estado = 20, 40 y 60 (cont.).

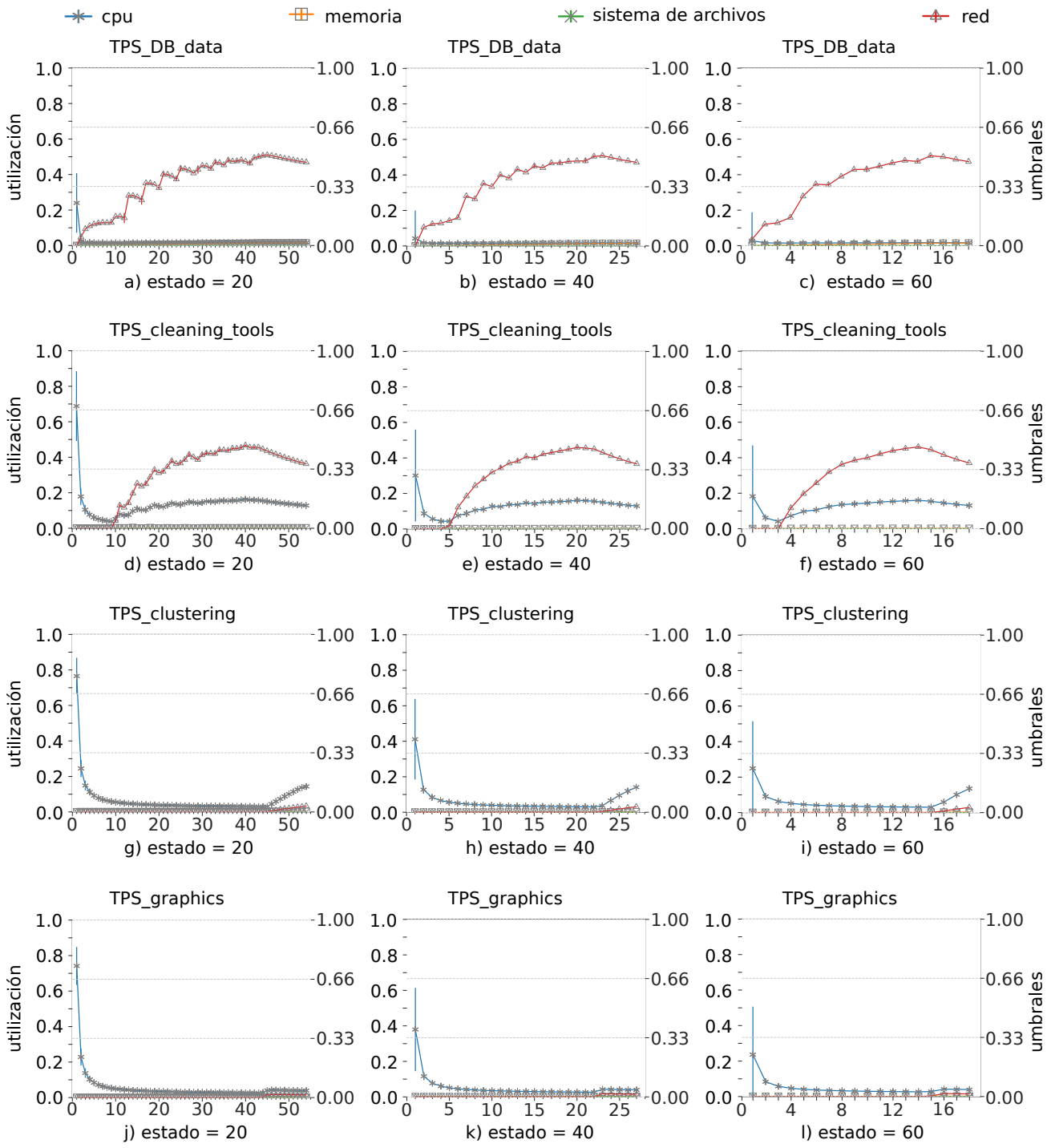


Figura 5.21: Resultados solución TPS con muestra = 10 y estado = 20, 40 y 60.

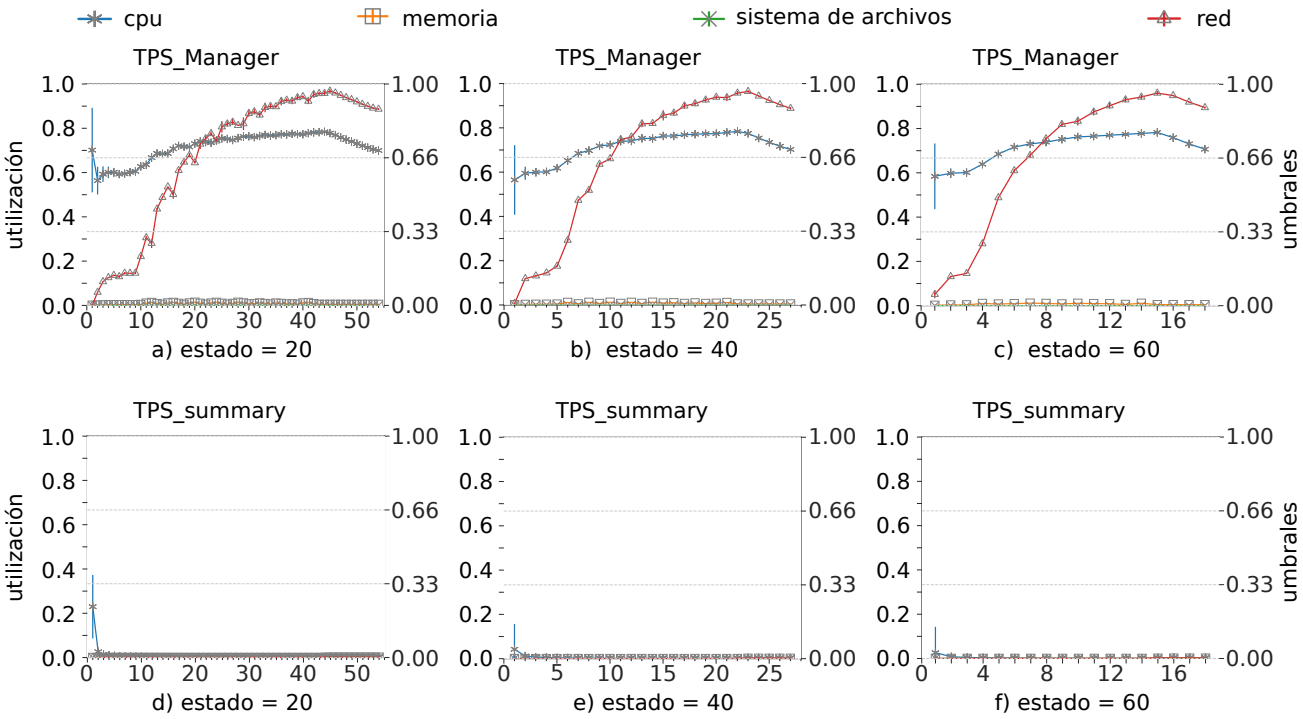


Figura 5.21: Resultados solución TPS con muestra = 10 y estado = 20, 40 y 60 (cont.).

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0032	0.0032	0.0032	0.0003	8.16E-09	0.0003	8.16E-09	0.0003	8.16E-09	5.91E-06	1.59E-05	1.59E-05
max	0.2602	0.2602	0.2602	0.0149	1.57E-07	0.0148	1.57E-07	0.0149	1.57E-07	0.5199	0.5187	0.5173
media	0.0172	0.0173	0.0176	0.0082	5.71E-08	0.0082	5.57E-08	0.0082	5.59E-08	0.3612	0.3590	0.3617
mediana	0.0162	0.0162	0.0163	0.0079	4.55E-08	0.0083	4.55E-08	0.0079	4.55E-08	0.4284	0.4307	0.4300
std	0.0113	0.0134	0.0160	0.0047	3.91E-08	0.0048	3.79E-08	0.0048	3.84E-08	0.1459	0.1506	0.1465
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	medio	medio	medio

Tabla 5.35: Estadísticos descriptivos de *DB_data* con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0129	0.0130	0.0130	0.0008	0.0008	0.0008	6.99E-09	6.99E-09	6.99E-09	1.97E-06	2.08E-06	2.05E-06
max	1	1	1	0.0025	0.0025	0.0023	8.39E-08	8.39E-08	8.39E-08	0.4751	0.4690	0.4649
media	0.1306	0.1307	0.1316	0.0017	0.0017	0.0017	7.91E-08	7.99E-08	7.87E-08	0.3000	0.3005	0.2998
mediana	0.1377	0.1380	0.1385	0.0019	0.0019	0.0018	8.28E-08	8.28E-08	8.28E-08	0.3753	0.3776	0.3719
std	0.0670	0.0704	0.0789	0.0004	0.0004	0.0004	1.73E-08	1.53E-08	1.80E-08	0.1636	0.1632	0.1626
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	medio	medio	medio

Tabla 5.36: Estadísticos descriptivos de *cleaning_tools* con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0263	0.0264	0.0264	0.0013	0.0013	0.0013	6.99E-09	6.99E-09	6.99E-09	1.46E-06	1.46E-06	1.47E-06
max	1	1	1	0.0038	0.0037	0.0036	4.55E-07	4.55E-07	4.55E-07	0.0332	0.0332	0.0314
media	0.0666	0.0664	0.0670	0.0017	0.0017	0.0017	3.80E-07	3.86E-07	3.79E-07	0.0034	0.0035	0.0034
mediana	0.0388	0.0388	0.0391	0.0015	0.0015	0.0015	3.95E-07	3.95E-07	3.95E-07	1.78E-06	1.80E-06	1.76E-06
std	0.0882	0.0854	0.0927	0.0005	0.0005	0.0005	8.47E-08	7.31E-08	8.78E-08	0.0082	0.0084	0.0082
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo

Tabla 5.37: Estadísticos descriptivos de *clustering* con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0231	0.0231	0.0232	0.0013	0.0013	0.0013	6.99E-09	6.99E-09	6.99E-09	1.45E-06	1.46E-06	1.46E-06
max	1	1	1	0.0025	0.0025	0.0025	2.21E-08	2.21E-08	2.21E-08	0.0178	0.0175	0.0175
media	0.0531	0.0528	0.0536	0.0016	0.0016	0.0016	1.68E-08	1.70E-08	1.68E-08	0.0030	0.0030	0.0030
mediana	0.0355	0.0355	0.0353	0.0014	0.0014	0.0014	1.63E-08	1.63E-08	1.63E-08	1.78E-06	1.79E-06	1.80E-06
std	0.0843	0.0849	0.0938	0.0004	0.0004	0.0004	3.01E-09	2.92E-09	3.11E-09	0.0062	0.0062	0.0060
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo

Tabla 5.38: Estadísticos descriptivos de *graphics* con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0929	0.1100	0.1100	0.0008	0.0008	0.0008	8.16E-09	8.16E-09	8.16E-09	1.40E-05	1.40E-05	1.40E-05
max	1	1	1	0.0163	0.0146	0.0124	2.95E-07	2.95E-07	2.95E-07	0.9863	0.9820	0.9800
media	0.7117	0.7131	0.7132	0.0071	0.0068	0.0072	2.40E-07	2.43E-07	2.39E-07	0.6746	0.6735	0.6751
mediana	0.7382	0.7382	0.7371	0.0052	0.0052	0.0085	2.47E-07	2.47E-07	2.47E-07	0.8240	0.8244	0.8266
std	0.0776	0.0744	0.0759	0.0038	0.0035	0.0031	5.30E-08	4.71E-08	5.49E-08	0.3088	0.3124	0.3088
escala de riesgo	alto	alto	alto	bajo	bajo	bajo	bajo	bajo	bajo	alto	alto	alto

Tabla 5.39: Estadísticos descriptivos de *Manager* con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	1.03E-05	1.03E-05	1.03E-05	0.0005	0.0005	0.0005	6.99E-09	6.99E-09	6.99E-09	1.44E-06	1.44E-06	1.44E-06
max	0.2245	0.2245	0.2245	0.0018	0.0018	0.0018	8.16E-09	8.16E-09	8.16E-09	0.0023	0.0023	0.0022
media	0.0037	0.0036	0.0038	0.0007	0.0007	0.0008	8.10E-09	8.12E-09	8.10E-09	0.0004	0.0004	0.0004
mediana	0.0017	0.0017	0.0017	0.0005	0.0005	0.0005	8.16E-09	8.16E-09	8.16E-09	1.76E-06	1.76E-06	1.79E-06
std	0.0109	0.0119	0.0135	0.0005	0.0005	0.0005	2.46E-10	2.14E-10	2.52E-10	0.0008	0.0008	0.0008
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo

Tabla 5.40: Estadísticos descriptivos de *summary* con muestra = 1.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0033	0.0033	0.0033	0.0003	0.0003	0.0003	8.16E-09	8.16E-09	8.16E-09	2.62E-06	1.73E-05	1.73E-05
max	0.6321	0.6321	0.6321	0.0149	0.0149	0.0148	1.57E-07	1.57E-07	1.57E-07	0.5191	0.5175	0.5173
media	0.0210	0.0190	0.0198	0.0082	0.0082	0.0081	5.57E-08	5.60E-08	5.59E-08	0.3598	0.3583	0.3586
mediana	0.0162	0.0162	0.0163	0.0079	0.0080	0.0083	4.55E-08	4.55E-08	4.55E-08	0.4266	0.4294	0.4294
std	0.0402	0.0349	0.0422	0.0047	0.0048	0.0048	3.79E-08	3.80E-08	3.84E-08	0.1481	0.1516	0.1509
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	medio	medio	medio

Tabla 5.41: Estadísticos descriptivos de *DB_data* con muestra = 10.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0129	0.0131	0.0130	0.0008	0.0008	0.0008	6.99E-09	6.99E-09	6.99E-09	1.95E-06	2.08E-06	2.03E-06
max	1	1	1	0.0055	0.0042	0.0024	8.39E-08	8.39E-08	8.39E-08	0.4684	0.4679	0.4667
media	0.1362	0.1335	0.1336	0.0020	0.0017	0.0017	7.91E-08	7.89E-08	7.87E-08	0.2989	0.2996	0.2987
mediana	0.1382	0.1380	0.1388	0.0019	0.0019	0.0018	8.28E-08	8.28E-08	8.28E-08	0.3761	0.3790	0.3726
std	0.0937	0.0801	0.0873	0.0010	0.0005	0.0004	1.73E-08	1.62E-08	1.80E-08	0.1642	0.1641	0.1629
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	medio	medio	medio

Tabla 5.42: Estadísticos descriptivos de *cleaning_tools* con muestra = 10.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0263	0.0265	0.0264	0.0006	0.0006	0.0006	6.99E-09	6.99E-09	6.99E-09	1.46E-06	1.46E-06	1.46E-06
max	1	1	1	0.0038	0.0037	0.0037	4.55E-07	4.55E-07	4.55E-07	0.0332	0.0332	0.0304
media	0.0716	0.0702	0.0691	0.0017	0.0017	0.0017	3.80E-07	3.79E-07	3.79E-07	0.0032	0.0034	0.0033
mediana	0.0387	0.0387	0.0393	0.0015	0.0015	0.0015	3.95E-07	3.95E-07	3.95E-07	1.79E-06	1.80E-06	1.77E-06
std	0.1120	0.0983	0.0989	0.0005	0.0005	0.0005	8.47E-08	7.93E-08	8.78E-08	0.0080	0.0082	0.0080
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo

Tabla 5.43: Estadísticos descriptivos de *clustering* con muestra = 10.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.0231	0.0231	0.0232	0.0007	0.0007	0.0007	6.99E-09	6.99E-09	6.99E-09	1.46E-06	1.46E-06	1.46E-06
max	1	1	1	0.0025	0.0025	0.0025	2.21E-08	2.21E-08	2.21E-08	0.0178	0.0176	0.0171
media	0.0584	0.0562	0.0555	0.0016	0.0016	0.0016	1.68E-08	1.68E-08	1.68E-08	0.0030	0.0030	0.0029
mediana	0.0352	0.0357	0.0349	0.0014	0.0014	0.0014	1.63E-08	1.63E-08	1.63E-08	1.79E-06	1.79E-06	1.80E-06
std	0.1088	0.0939	0.0967	0.0004	0.0004	0.0004	3.01E-09	2.92E-09	3.11E-09	0.0062	0.0062	0.0060
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo

Tabla 5.44: Estadísticos descriptivos de *graphics* con muestra = 10.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0.1805	0.3462	0.3846	0.0007	0.0008	0.0008	8.16E-09	8.16E-09	8.16E-09	2.94E-05	3.28E-05	3.28E-05
max	1	1	1	0.0160	0.0143	0.0139	2.95E-07	2.95E-07	2.95E-07	0.9844	0.9796	0.9796
media	0.7145	0.7132	0.7137	0.0071	0.0070	0.0071	2.40E-07	2.40E-07	2.39E-07	0.6714	0.6710	0.6707
mediana	0.7384	0.7389	0.7355	0.0056	0.0061	0.0074	2.47E-07	2.47E-07	2.47E-07	0.8167	0.8212	0.8268
std	0.0715	0.0734	0.0744	0.0036	0.0034	0.0031	5.30E-08	5.00E-08	5.49E-08	0.3115	0.3147	0.3130
escala de riesgo	alto	alto	alto	bajo	bajo	bajo	bajo	bajo	bajo	alto	alto	alto

Tabla 5.45: Estadísticos descriptivos de *Manager* con muestra = 10.

Estadístico	CPU			memoria			sistema de archivos			red		
	20	40	60	20	40	60	20	40	60	20	40	60
# de datos	1455	733	490	1455	733	490	1455	733	490	1455	733	490
min	0	0	0	0.0005	0.0005	0.0005	6.99E-09	6.99E-09	6.99E-09	1.43E-06	1.44E-06	1.44E-06
max	0.5264	0.5264	0.5264	0.0019	0.0018	0.0018	8.16E-09	8.16E-09	8.16E-09	0.0022	0.0022	0.0022
media	0.0072	0.0047	0.0050	0.0007	0.0007	0.0007	8.10E-09	8.10E-09	8.10E-09	0.0004	0.0004	0.0004
mediana	0.0017	0.0017	0.0017	0.0005	0.0005	0.0005	8.16E-09	8.16E-09	8.16E-09	1.76E-06	1.76E-06	1.80E-06
std	0.0378	0.0261	0.0311	0.0005	0.0005	0.0005	2.46E-10	2.27E-10	2.52E-10	0.0008	0.0008	0.0008
escala de riesgo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo	bajo

Tabla 5.46: Estadísticos descriptivos de *summary* con muestra = 10.

5.7.4 Resultados de la representación

Para la representación del TPS se generaron fichas de representación tomando como base las características de un *Thing Description* (TD) del WoT. Un TD es utilizado como bloque de construcción central al que se le agrega la mayor información posible de un objeto *Thing*. Un Thing es una abstracción de una entidad física o virtual que tiene descritas sus interfaces y sus metadatos dentro de un TD. Tiene cuatro componentes principales: metadatos (del Thing), interacciones (indicando cómo puede ser usado el Thing), vocabulario (conteniendo definiciones utilizadas dentro del TD, que facilitan el entendimiento a los computadores) y enlaces Web (para representar las relaciones entre Things u otros documentos en la Web). Se distinguen tres tipos de interacciones propiedades, acciones y eventos. En las propiedades se expone información del Thing como es el caso del comportamiento (factores de utilización) y la estructura (entradas y salidas del contenedor). Las acciones se refieren a las funcionalidades del Thing por lo que en este apartado se incluyen las funciones que realizan los contenedores. Los eventos pueden ser alertas, en este caso las escalas de riesgo son agregadas aquí. La ficha corresponde a una estructura con formato JSON-LD[2] (del inglés JavaScript Object Notation for Linked Data) como lo recomienda el W3C[3]. Se generó una ficha por cada contenedor del Gestor, los cuales se muestran en el Anexo A.

[2]https://www.w3.org/TR/json-ld11/
[3]https://www.w3.org/

6

Conclusiones, limitaciones y trabajo futuro

En este capítulo se presentan las conclusiones generales del trabajo de investigación realizado, así como las contribuciones logradas. En la sección 6.2 se muestran las limitaciones identificadas de acuerdo a las dificultades presentadas a lo largo de la tesis. Finalmente, en la sección 6.3 se presentan las posibles direcciones que puede tomar esta tesis como trabajo futuro.

6.1 Conclusiones

En este trabajo de investigación se abordó el problema de la supervisión de sistemas de contenedores virtuales, los cuales se pueden ver afectados por factores externos e internos (de los contenedores) produciendo una interrupción del servicio o actividad que éstos realicen, por lo que es de interés conocer el punto de falla. Esto es, saber el(los) contenedor(es) involucrado(s).

Para dar solución a este problema, se presentó el diseño, desarrollo e implementación de un método de supervisión de contenedores virtuales. El método recibe como insumo un archivo de configuración de una solución de software que emplea contenedores virtuales. El método crea una

representación del estado de los contenedores virtuales basándose en escalas de riesgo de tres rangos, los cuales pueden ser consultados desde la ficha WoT. El método está compuesto de cuatro etapas: monitoreo, indexado, representación y diagnóstico. Mediante estas etapas se logra la supervisión dinámica de los componentes de una solución.

De acuerdo a la evaluación experimental mostrada en el Capítulo 5, se han encontrado algunos puntos que son mencionados a continuación. Mediante el uso de las fórmulas de utilización el prototipo de supervisión es capaz de mostrar los cambios en los estados de riesgo de las soluciones evaluadas. Estos cambios de estado se ven reflejados en los valores de utilización de cada métrica que van desde aplicar la carga mínima, e incrementarla hasta lograr la mayor carga.

La carga máxima lograda, así como los tiempos necesarios para detectar los cambios de estado se ven afectados por diversos factores. Unos de estos factores son, el tipo de procesamiento que realice la solución, así como el tamaño de archivos que sean procesados.

También se ha detectado que ejecutar una solución que realice una carga demandante en el mismo equipo sobre el que se ejecuta el prototipo tiene efectos negativos sobre la obtención de las métricas. Esto debido al encolamiento generado por el uso intensivo de los procesos que ejecute la solución, lo que puede retrasar la obtención de las métricas e incluso la pérdida de algunas de ellas. Para solventar este problema se puede colocar el prototipo en un equipo diferente aligerando la carga y realizando la supervisión desde un equipo externo. Esto es posible gracias a la modularización de los componentes del prototipo.

Las configuraciones de los tiempos de ventana, estado y muestra permiten comprender cómo varía el comportamiento capturado con cada una de ellas y cómo esto afecta o beneficia el resultado esperado. Para una solución ejecutándose por períodos cortos de tiempo se puede optar por la configuración de muestra = 1, estado = 20 y ventana = 120. Para una solución ejecutándose por períodos prolongados de tiempo se puede optar por la configuración de muestra = 10, estado = 60 y ventana = 1200. Para la mayoría de los casos una configuración intermedia es buena, la cual es muestra = 10, estado = 40 y ventana = 120.

Para cumplir con el primer objetivo específico: *"Desarrollar un método de monitoreo de variables de comportamiento dinámico de un SECI."* Se desarrolló un método de monitoreo el cual fue implementado como un agente desacoplado y consumible que provee las variables: CPU, memoria, sistema de archivos y red por contenedor.

Respecto al segundo objetivo específico: *"Definir un esquema de estructuras de cubo de datos para gestionar la supervisión de las propiedades de confiabilidad y eficiencia de los componentes de un SECI."* Se realizó un esquema basado en la estructura de cubo de datos para gestionar las propiedades de confiabilidad y eficiencia, el cual fue implementado como un servicio de indexado que almacena la información (en una base de datos NoSQL) y la pone a disposición de los otros módulos como documentos JSON.

Para lograr el tercer objetivo específico: *"Definir una representación abstracta del análisis del comportamiento dinámico de las variables de un SECI."* Se realizó una representación abstracta basada en multi-modelado, la cual fue generada siguiendo los lineamientos de WoT y contiene la información de estado de cada componente. Ésta representación denominada ficha se encuentra en formato JSON-LD como lo establece el WoT.

Al cumplir estos tres objetivos específicos se logró satisfacer el objetivo general logrando la supervisión dinámica de los componentes de un SECI mediante una representación abstracta materializada en una ficha de WoT. Esto permitió también corroborar la hipótesis, ya que se pudo supervisar el estado actual de un SECI mediante la indexación de sus variables de comportamiento (CPU, Memoria, Sistema de archivos y red) de las propiedades de confiabilidad y eficiencia.

6.2 Limitaciones

Si bien con la implementación del prototipo del método propuesto se han obtenido resultados alentadores que abren la puerta a seguir explorando con el método, durante el diseño e implementación del método se identificaron las siguientes limitaciones:

- La versión actual del prototipo realiza la comunicación entre módulos mediante API REST, no se consideró otro tipo de comunicación, por ejemplo, memoria compartida. Esto presenta un área de oportunidad debido a que si se despliega más de un módulo en el mismo equipo, éstos podrían comunicarse mediante memoria, desviando el uso de la red y mejorando su rendimiento.

- Para el correcto funcionamiento del prototipo se requiere como entrada un archivo de configuración de una solución basada de contenedores, usualmente en un formato conocido (YAML). Sin embargo, hacer uso de un archivo no es la única manera para desplegar contenedores. Por lo tanto, los contenedores desplegados de forma manual, individualmente mediante la consola de comandos, no se podrán supervisar con la versión actual del prototipo a menos que sean explícitamente declarados en un archivo de configuración.

6.3 Trabajo futuro

A partir de la limitaciones descritas anteriormente para el prototipo y las posibilidades que se tienen de extender el trabajo realizado, se podrían considerar los siguientes puntos como modificaciones y adaptaciones del método::

- Agregar atributos no funcionales a los componentes del método, por ejemplo, tolerancia a fallos y patrones de paralelismo.

- Permitir la compatibilidad con más agentes extractores de métricas, así como diferentes tipos de almacenamiento, por ejemplo, bases de datos de series de tiempo.

- Realizar la evaluación y migración de los contenedores y/o soluciones que se encuentren en el límite de uso de recursos hacia un entorno que les favorezca de acuerdo a sus necesidades.

A

Fichas generadas por la representación

En este capitulo se presentan las fichas que se generaron como producto de la representación de multi-modelado, las cuales forman parte de los resultados del Capitulo 5.

A.1 Fichas del estudio de caso

A continuación se muestran las fichas correspondientes a los contenedores del caso de estudio mostrados en la sección 5.7. La estructura de dichas fichas se describió en la sección 5.7.4.

Listado A.1: Ficha del contenedor DB_data.

```
{
  "@context": [
    "https://www.w3.org/2019/wot/td/v1",
    {
      "@version": 1.1,
      "@language": "en",
      "ctv": "https://www.tamps.cinvestav.mx/schema/",
```

```
      "utilization": {
        "@id": "ctv:util",
        "@type": "jsonschema:NumberSchema"
      }
    }
  ],
  "id": "http://www.tamps.cinvestav.mx/2
    ecceb547dff8132b5e6ecbd8b77ee5eac0ff7b88cc7a0a5c6ab073cc441f0e2",
  "@type": "Thing",
  "td:description": {
    "@value": "Database of tps manager"
  },
  "actions": {
    "@none": {}
  },
  "events": {
    "@none": {
      "ctv:CPUHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_hth"
          }
        }
      },
      "ctv:CPULowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_lth"
```

```
        }
      }
    },
    "ctv:CPUMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_mth"
        }
      }
    },
    "ctv:FSHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_hth"
        }
      }
    },
    "ctv:FSLowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_lth"
        }
      }
    },
    "ctv:FSMediumThreshold": {
```

```
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_mth"
          }
        }
      },
      "ctv:MEMHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_hth"
          }
        }
      },
      "ctv:MEMLowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_lth"
          }
        }
      },
      "ctv:MEMMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
```

```
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_mth"
          }
        }
      },
      "ctv:NETHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_hth"
          }
        }
      },
      "ctv:NETLowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_lth"
          }
        }
      },
      "ctv:NETMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_mth"
          }
        }
```

```
      }
    }
  },
  "properties": {
    "@none": {
      "ctv:behavior": {
        "@type": "string",
        "ctv:cpu_util": {
          "@type": "number",
          "ctv:util": 0.0162,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the cpu utilization of the Thing"
          }
        },
        "ctv:filesystem_util": {
          "@type": "number",
          "ctv:util": 0.0079,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the filesystem utilization of the Thing"
          }
        },
        "ctv:memory_util": {
          "@type": "number",
          "ctv:util": 0.0000000455,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the memory utilization of the Thing"
          }
        },
        "ctv:network_util": {
          "@type": "number",
          "ctv:util": 0.4307,
```

```
177           "jsonschema:readOnly": true,
178           "td:description": {
179             "@value": "Shows the network utilization of the Thing"
180           }
181         },
182         "td:description": {
183           "@value": "Shows params about the behavior of the Thing"
184         }
185       },
186       "ctv:structure": {
187         "@type": "string",
188         "td:description": {
189           "@value": "Shows params about the structure of the Thing"
190         },
191         "input": {
192           "@type": "string",
193           "td:description": {
194             "@value": "Base de datos de datos procesados por TPS."
195           }
196         },
197         "output": {
198           "@type": "string",
199           "td:description": {
200             "@value": "Base de datos de datos procesados por TPS."
201           }
202         }
203       }
204     }
205   },
206   "td:title": {
207     "@value": "TPS_DB_data"
208   }
209 }
```

Listado A.2: Ficha del contenedor Cleaning_tools.

```
{
  "@context": [
    "https://www.w3.org/2019/wot/td/v1",
    {
      "@version": 1.1,
      "@language": "en",
      "ctv": "https://www.tamps.cinvestav.mx/schema/",
      "utilization": {
        "@id": "ctv:util",
        "@type": "jsonschema:NumberSchema"
      }
    }
  ],
  "id": "http://www.tamps.cinvestav.mx/
    e6e3f3ed08360f65d617a7aae904f8f5da765aeb274fd05b0c79a77cd2cfdb1c",
  "@type": "Thing",
  "td:description": {
    "@value": "Tools to clean data"
  },
  "actions": {
    "@none": {
      "ctv:cleaningtools": {
        "td:forms": {
          "td:description": {
            "@value": "Aplica una serie de funciones para eliminacion de ruido."
          },
          "td:href": {
            "@value": "POST http://localhost:11003/cleaningtools"
          }
        }
      },
      "ctv:transform": {
```

```
      "td:forms": {
        "td:description": {
          "@value": "Transforma la estructura de un dataset de filas a columnas."
        },
        "td:href": {
          "@value": "POST http://localhost:11003/transform"
        }
      }
    }
  }
},
"events": {
  "@none": {
    "ctv:CPUHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_hth"
        }
      }
    },
    "ctv:CPULowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_lth"
        }
      }
    },
    "ctv:CPUMediumThreshold": {
```

```
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_mth"
        }
      }
    },
    "ctv:FSHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_hth"
        }
      }
    },
    "ctv:FSLowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_lth"
        }
      }
    },
    "ctv:FSMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
```

```
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_mth"
          }
        }
      },
      "ctv:MEMHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_hth"
          }
        }
      },
      "ctv:MEMLowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_lth"
          }
        }
      },
      "ctv:MEMMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_mth"
          }
        }
```

```
    },
    "ctv:NETHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/net_hth"
        }
      }
    },
    "ctv:NETLowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/net_lth"
        }
      }
    },
    "ctv:NETMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/net_mth"
        }
      }
    }
  }
},
"properties": {
```

```
    "@none": {
      "ctv:behavior": {
        "@type": "string",
        "ctv:cpu_util": {
          "@type": "number",
          "ctv:util": 0.138,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the cpu utilization of the Thing"
          }
        },
        "ctv:filesystem_util": {
          "@type": "number",
          "ctv:util": 0.0000000828.,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the filesystem utilization of the Thing"
          }
        },
        "ctv:memory_util": {
          "@type": "number",
          "ctv:util": 0.0019,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the memory utilization of the Thing"
          }
        },
        "ctv:network_util": {
          "@type": "number",
          "ctv:util": 0.3776,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the network utilization of the Thing"
          }
```

```
202         },
203         "td:description": {
204           "@value": "Shows params about the behavior of the Thing"
205         }
206       },
207       "ctv:structure": {
208         "@type": "string",
209         "td:description": {
210           "@value": "Shows params about the structure of the Thing"
211         },
212         "input": {
213           "@type": "string",
214           "ctv:method": {
215             "@value": "{http_method}"
216           },
217           "ctv:query": {
218             "@value": "http://localhost:11003/{option}"
219           },
220           "td:description": {
221             "@value": "Dataset y parametros"
222           }
223         },
224         "output": {
225           "@type": "string",
226           "ctv:response": {
227             "@value": "{response}"
228           },
229           "td:description": {
230             "@value": "Dataset transformado."
231           }
232         }
233       }
234     }
235   },
```

```
  "td:title": {
    "@value": "TPS_cleaning_tools"
  }
}
```

Listado A.3: Ficha del contenedor clustering.

```
{
  "@context": [
    "https://www.w3.org/2019/wot/td/v1",
    {
      "@version": 1.1,
      "@language": "en",
      "ctv": "https://www.tamps.cinvestav.mx/schema/",
      "utilization": {
        "@id": "ctv:util",
        "@type": "jsonschema:NumberSchema"
      }
    }
  ],
  "id": "http://www.tamps.cinvestav.mx/
    c7c517fc4315b77da662b9ea35b5a8a204bfbb2320f77a9ca257b57dbc44a1dd",
  "@type": "Thing",
  "td:description": {
    "@value": "API Gateway container"
  },
  "actions": {
    "@none": {
      "ctv:hierarchical": {
        "td:forms": {
          "td:description": {
            "@value": "Realiza el agrupamiento de los datos utilizando un algoritmo jerarquico
    ."
          },
          "td:href": {
```

```
27            "@value": "POST http://localhost:11001/herarhical"
28          }
29        }
30      },
31      "ctv:kmeans": {
32        "td:forms": {
33          "td:description": {
34            "@value": "Realiza el agrupamiento de los datos utilizando el algoritmo kmeans. Se
    agrega una etiqueta de clase (class) a cada registro."
35          },
36          "td:href": {
37            "@value": "POST http://localhost:11001/kmeans"
38          }
39        }
40      },
41      "ctv:silhouette": {
42        "td:forms": {
43          "td:description": {
44            "@value": "Realiza una comparativa entre los algoritmos de kmeans y jerarquico con
    el metodo single, comprobando valores de k de 1 a 15. El resultado de las pruebas se
    grafica."
45          },
46          "td:href": {
47            "@value": "POST http://localhost:11001/silhouette"
48          }
49        }
50      }
51    }
52  },
53  "events": {
54    "@none": {
55      "ctv:CPUHighThreshold": {
56        "td:data": {
57          "@type": "bool"
```

```
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_hth"
        }
      }
    },
    "ctv:CPULowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_lth"
        }
      }
    },
    "ctv:CPUMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_mth"
        }
      }
    },
    "ctv:FSHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_hth"
```

```
        }
      }
    },
    "ctv:FSLowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_lth"
        }
      }
    },
    "ctv:FSMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_mth"
        }
      }
    },
    "ctv:MEMHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_hth"
        }
      }
    },
    "ctv:MEMLowThreshold": {
```

```
126         "td:data": {
127           "@type": "bool"
128         },
129         "td:forms": {
130           "td:href": {
131             "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_lth"
132           }
133         }
134       },
135       "ctv:MEMMediumThreshold": {
136         "td:data": {
137           "@type": "bool"
138         },
139         "td:forms": {
140           "td:href": {
141             "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_mth"
142           }
143         }
144       },
145       "ctv:NETHighThreshold": {
146         "td:data": {
147           "@type": "bool"
148         },
149         "td:forms": {
150           "td:href": {
151             "@value": "https://www.tamps.cinvestav.mx/supervisor/net_hth"
152           }
153         }
154       },
155       "ctv:NETLowThreshold": {
156         "td:data": {
157           "@type": "bool"
158         },
159         "td:forms": {
```

```
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/net_lth"
        }
      }
    },
    "ctv:NETMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/net_mth"
        }
      }
    }
  }
},
"properties": {
  "@none": {
    "ctv:behavior": {
      "@type": "string",
      "ctv:cpu_util": {
        "@type": "number",
        "ctv:util": 0.0388,
        "jsonschema:readOnly": true,
        "td:description": {
          "@value": "Shows the cpu utilization of the Thing"
        }
      },
      "ctv:filesystem_util": {
        "@type": "number",
        "ctv:util": 0.000000395,
        "jsonschema:readOnly": true,
        "td:description": {
```

```
            "@value": "Shows the filesystem utilization of the Thing"
          }
        },
        "ctv:memory_util": {
          "@type": "number",
          "ctv:util": 0.0015,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the memory utilization of the Thing"
          }
        },
        "ctv:network_util": {
          "@type": "number",
          "ctv:util": 0.0000018,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the network utilization of the Thing"
          }
        },
        "td:description": {
          "@value": "Shows params about the behavior of the Thing"
        }
      },
      "ctv:structure": {
        "@type": "string",
        "td:description": {
          "@value": "Shows params about the structure of the Thing"
        },
        "input": {
          "@type": "string",
          "ctv:method": {
            "@value": "{http_method}"
          },
          "ctv:query": {
```

```
            "@value": "http://localhost:11001/api/v1/{option}"
          },
          "td:description": {
            "@value": "Dataset y parametros de clustering."
          }
        },
        "output": {
          "@type": "string",
          "ctv:response": {
            "@value": "{response}"
          },
          "td:description": {
            "@value": "Dataset etiquetado."
          }
        }
      }
    }
  },
  "td:title": {
    "@value": "TPS_clustering"
  }
}
```

Listado A.4: Ficha del contenedor graphics.

```
{
  "@context": [
    "https://www.w3.org/2019/wot/td/v1",
    {
      "@version": 1.1,
      "@language": "en",
      "ctv": "https://www.tamps.cinvestav.mx/schema/",
      "utilization": {
        "@id": "ctv:util",
        "@type": "jsonschema:NumberSchema"
```

```
      }
    }
  ],
  "id": "http://www.tamps.cinvestav.mx/8
    c281f4be6e5a55cd110f501f6682b4f9f9bad23d1ff21bf6c859f3e99ba736a",
  "@type": "Thing",
  "td:description": {
    "@value": "service to graphic data"
  },
  "actions": {
    "@none": {
      "ctv:bar": {
        "td:forms": {
          "td:description": {
            "@value": "Genera una grafica de barras"
          },
          "td:href": {
            "@value": "POST http://localhost:3132/hist"
          }
        }
      },
      "ctv:get_images": {
        "td:forms": {
          "td:description": {
            "@value": "Retorna un archivo en formato ZIP con las imagenes generadas."
          },
          "td:href": {
            "@value": "POST http://localhost:3132/get_images"
          }
        }
      },
      "ctv:hist": {
        "td:forms": {
          "td:description": {
```

```
44            "@value": "Genera un histograma."
45          },
46          "td:href": {
47            "@value": "POST http://localhost:3132/cleaningtools"
48          }
49        }
50      },
51      "ctv:line": {
52        "td:forms": {
53          "td:description": {
54            "@value": "Genera una grafica linear"
55          },
56          "td:href": {
57            "@value": "POST http://localhost:3132/line"
58          }
59        }
60      },
61      "ctv:scatter": {
62        "td:forms": {
63          "td:description": {
64            "@value": "Genera una grafica de puntos."
65          },
66          "td:href": {
67            "@value": "POST http://localhost:3132/scatter"
68          }
69        }
70      }
71    }
72  },
73  "events": {
74    "@none": {
75      "ctv:CPUHighThreshold": {
76        "td:data": {
77          "@type": "bool"
```

```
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_hth"
        }
      }
    },
    "ctv:CPULowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_lth"
        }
      }
    },
    "ctv:CPUMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_mth"
        }
      }
    },
    "ctv:FSHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_hth"
```

```
        }
      }
    },
    "ctv:FSLowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_lth"
        }
      }
    },
    "ctv:FSMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_mth"
        }
      }
    },
    "ctv:MEMHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_hth"
        }
      }
    },
    "ctv:MEMLowThreshold": {
```

```
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_lth"
          }
        }
      },
      "ctv:MEMMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_mth"
          }
        }
      },
      "ctv:NETHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_hth"
          }
        }
      },
      "ctv:NETLowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
```

```
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_lth"
          }
        }
      },
      "ctv:NETMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_mth"
          }
        }
      }
    }
  },
  "properties": {
    "@none": {
      "ctv:behavior": {
        "@type": "string",
        "ctv:cpu_util": {
          "@type": "number",
          "ctv:util": 0.0355,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the cpu utilization of the Thing"
          }
        },
        "ctv:filesystem_util": {
          "@type": "number",
          "ctv:util": 0.0000000163,
          "jsonschema:readOnly": true,
          "td:description": {
```

```
            "@value": "Shows the filesystem utilization of the Thing"
          }
        },
        "ctv:memory_util": {
          "@type": "number",
          "ctv:util": 0.0014,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the memory utilization of the Thing"
          }
        },
        "ctv:network_util": {
          "@type": "number",
          "ctv:util": 0.00000179,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the network utilization of the Thing"
          }
        },
        "td:description": {
          "@value": "Shows params about the behavior of the Thing"
        }
      },
      "ctv:structure": {
        "@type": "string",
        "td:description": {
          "@value": "Shows params about the structure of the Thing"
        },
        "input": {
          "@type": "string",
          "ctv:method": {
            "@value": "{http_method}"
          },
          "ctv:query": {
```

```
            "@value": "http://localhost:3132/{option}"
          },
          "td:description": {
            "@value": "Dataset y parametros"
          }
        },
        "output": {
          "@type": "string",
          "ctv:response": {
            "@value": "{Binary data}"
          },
          "td:description": {
            "@value": "Retorna un archivo binario con base en el metodo utilizado"
          }
        }
      }
    }
  },
  "td:title": {
    "@value": "TPS_graphics"
  }
}
```

Listado A.5: Ficha del contenedor Manager.

```
{
  "@context": [
    "https://www.w3.org/2019/wot/td/v1",
    {
      "@version": 1.1,
      "@language": "en",
      "ctv": "https://www.tamps.cinvestav.mx/schema/",
      "utilization": {
        "@id": "ctv:util",
        "@type": "jsonschema:NumberSchema"
```

```
      }
    }
  ],
  "id": "http://www.tamps.cinvestav.mx/5
    a9d47c5c1e36124f87cf985ae21e51bb1662fee1407d8ca201fc9fd84232893",
  "@type": "Thing",
  "td:description": {
    "@value": "API Gateway and the manager"
  },
  "actions": {
    "@none": {
      "ctv:check": {
        "td:forms": {
          "td:description": {
            "@value": "verifica si un servicio se encuentra activo o no"
          },
          "td:href": {
            "@value": "GET http://localhost:54350/check/{service}"
          }
        }
      },
      "ctv:getData": {
        "td:forms": {
          "td:description": {
            "@value": "devuelve los datos de un workspace y un datasource del mismo, esto en
    formato json."
          },
          "td:href": {
            "@value": "GET http://localhost:54350/{workspace}/{datasource}/"
          }
        }
      },
      "ctv:putData": {
        "td:forms": {
```

```
43          "td:description": {
44            "@value": "coloca datos en un workspace"
45          },
46          "td:href": {
47            "@value": "GET http://localhost:54350/{workspace}/putdata/{datasource}/"
48          }
49        }
50      }
51    }
52  },
53  "events": {
54    "@none": {
55      "ctv:CPUHighThreshold": {
56        "td:data": {
57          "@type": "bool"
58        },
59        "td:forms": {
60          "td:href": {
61            "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_hth"
62          }
63        }
64      },
65      "ctv:CPULowThreshold": {
66        "td:data": {
67          "@type": "bool"
68        },
69        "td:forms": {
70          "td:href": {
71            "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_lth"
72          }
73        }
74      },
75      "ctv:CPUMediumThreshold": {
76        "td:data": {
```

```
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_mth"
          }
        }
      },
      "ctv:FSHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_hth"
          }
        }
      },
      "ctv:FSLowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_lth"
          }
        }
      },
      "ctv:FSMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
```

```
            "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_mth"
          }
        }
      },
      "ctv:MEMHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_hth"
          }
        }
      },
      "ctv:MEMLowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_lth"
          }
        }
      },
      "ctv:MEMMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_mth"
          }
        }
      },
```

```
      "ctv:NETHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_hth"
          }
        }
      },
      "ctv:NETLowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_lth"
          }
        }
      },
      "ctv:NETMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_mth"
          }
        }
      }
    }
  },
  "properties": {
    "@none": {
```

```
    "ctv:behavior": {
      "@type": "string",
      "ctv:cpu_util": {
        "@type": "number",
        "ctv:util": 0.7382,
        "jsonschema:readOnly": true,
        "td:description": {
          "@value": "Shows the cpu utilization of the Thing"
        }
      },
      "ctv:filesystem_util": {
        "@type": "number",
        "ctv:util": 0.000000247,
        "jsonschema:readOnly": true,
        "td:description": {
          "@value": "Shows the filesystem utilization of the Thing"
        }
      },
      "ctv:memory_util": {
        "@type": "number",
        "ctv:util": 0.0052,
        "jsonschema:readOnly": true,
        "td:description": {
          "@value": "Shows the memory utilization of the Thing"
        }
      },
      "ctv:network_util": {
        "@type": "number",
        "ctv:util": 0.8244,
        "jsonschema:readOnly": true,
        "td:description": {
          "@value": "Shows the network utilization of the Thing"
        }
      },
```

```
213         "td:description": {
214           "@value": "Shows params about the behavior of the Thing"
215         }
216       },
217       "ctv:structure": {
218         "@type": "string",
219         "td:description": {
220           "@value": "Shows params about the structure of the Thing"
221         },
222         "input": {
223           "@type": "string",
224           "ctv:method": {
225             "@value": "{http_method}"
226           },
227           "ctv:query": {
228             "@value": "http://localhost:54350/api/v1/{option}"
229           },
230           "td:description": {
231             "@value": "Shows the input of the Thing"
232           }
233         },
234         "output": {
235           "@type": "string",
236           "ctv:response": {
237             "@value": "{response}"
238           },
239           "td:description": {
240             "@value": "Shows the output of the Thing"
241           }
242         }
243       }
244     }
245   },
246   "td:title": {
```

```
    "@value": "TPS_Manager"
  }
}
```

Listado A.6: Ficha del contenedor summary.

```
{
  "@context": [
    "https://www.w3.org/2019/wot/td/v1",
    {
      "@version": 1.1,
      "@language": "en",
      "ctv": "https://www.tamps.cinvestav.mx/schema/",
      "utilization": {
        "@id": "ctv:util",
        "@type": "jsonschema:NumberSchema"
      }
    }
  ],
  "id": "http://www.tamps.cinvestav.mx/96
    d9d0e1c315d12141c36f0a89489f9a0a9dc925f7faf93e382757b5f5ebad28",
  "@type": "Thing",
  "td:description": {
    "@value": "TPS para la obtencion de datos estadisticos programado en lenguaje R, el cual
    puede ser accedido a traves de peticiones rest."
  },
  "actions": {
    "@none": {
      "ctv:correlation": {
        "td:forms": {
          "td:description": {
            "@value": "Obtiene la varianza, covarianza, y coeficiente de correlacion de un
    dataset."
          },
          "td:href": {
```

```
27            "@value": "POST http://localhost:11002/api/v1/correlation"
28          }
29        }
30      },
31      "ctv:covariance": {
32        "td:forms": {
33          "td:description": {
34            "@value": "Calcula la covarianza de al menos 2 variables dentro de un dataset."
35          },
36          "td:href": {
37            "@value": "POST http://localhost:11002/api/v1/covariance"
38          }
39        }
40      },
41      "ctv:describe": {
42        "td:forms": {
43          "td:description": {
44            "@value": "Realiza una descripcion estadistica de un dataset mediante el calculo de
    medidas de tendencia central."
45          },
46          "td:href": {
47            "@value": "POST http://localhost:11002/api/v1/describe"
48          }
49        }
50      }
51    }
52  },
53  "events": {
54    "@none": {
55      "ctv:CPUHighThreshold": {
56        "td:data": {
57          "@type": "bool"
58        },
59        "td:forms": {
```

```
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_hth"
          }
        }
      },
      "ctv:CPULowThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_lth"
          }
        }
      },
      "ctv:CPUMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/cpu_mth"
          }
        }
      },
      "ctv:FSHighThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_hth"
          }
        }
```

```
    },
    "ctv:FSLowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_lth"
        }
      }
    },
    "ctv:FSMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/fs_mth"
        }
      }
    },
    "ctv:MEMHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_hth"
        }
      }
    },
    "ctv:MEMLowThreshold": {
      "td:data": {
        "@type": "bool"
```

```
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_lth"
        }
      }
    },
    "ctv:MEMMediumThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/mem_mth"
        }
      }
    },
    "ctv:NETHighThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/net_hth"
        }
      }
    },
    "ctv:NETLowThreshold": {
      "td:data": {
        "@type": "bool"
      },
      "td:forms": {
        "td:href": {
          "@value": "https://www.tamps.cinvestav.mx/supervisor/net_lth"
```

```
          }
        }
      },
      "ctv:NETMediumThreshold": {
        "td:data": {
          "@type": "bool"
        },
        "td:forms": {
          "td:href": {
            "@value": "https://www.tamps.cinvestav.mx/supervisor/net_mth"
          }
        }
      }
    }
  },
  "properties": {
    "@none": {
      "ctv:behavior": {
        "@type": "string",
        "ctv:cpu_util": {
          "@type": "number",
          "ctv:util": 0.0017,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the cpu utilization of the Thing"
          }
        },
        "ctv:filesystem_util": {
          "@type": "number",
          "ctv:util": 0.00000000816,
          "jsonschema:readOnly": true,
          "td:description": {
            "@value": "Shows the filesystem utilization of the Thing"
          }
```

```
196         },
197         "ctv:memory_util": {
198           "@type": "number",
199           "ctv:util": 0.0005,
200           "jsonschema:readOnly": true,
201           "td:description": {
202             "@value": "Shows the memory utilization of the Thing"
203           }
204         },
205         "ctv:network_util": {
206           "@type": "number",
207           "ctv:util": 0.00000176,
208           "jsonschema:readOnly": true,
209           "td:description": {
210             "@value": "Shows the network utilization of the Thing"
211           }
212         },
213         "td:description": {
214           "@value": "Shows params about the behavior of the Thing"
215         }
216       },
217       "ctv:structure": {
218         "@type": "string",
219         "td:description": {
220           "@value": "Shows params about the structure of the Thing"
221         },
222         "input": {
223           "@type": "string",
224           "ctv:method": {
225             "@value": "{http_method}"
226           },
227           "ctv:query": {
228             "@value": "http://localhost:11002/api/v1/{option}"
229           },
```

```
230           "td:description": {
231             "@value": "Dataset y parametros de funciones estadisticas."
232           }
233         },
234         "output": {
235           "@type": "string",
236           "ctv:response": {
237             "@value": "{response}"
238           },
239           "td:description": {
240             "@value": "Resumen en formato JSON describiendo el dataset."
241           }
242         }
243       }
244     }
245   },
246   "td:title": {
247     "@value": "TPS_summary"
248   }
249 }
```

Bibliografía

[1] Armenise, V. (2015). Continuous delivery with jenkins: Jenkins solutions to implement continuous delivery. In *Proceedings of the Third International Workshop on Release Engineering, 19-19 May*, RELENG '15, pages 24–27, Piscataway, NJ, USA. IEEE Press.

[2] Avgeriou, P. and Zdun, U. (2005). Architectural patterns revisited - a pattern language. In *EuroPLoP' 2005, Tenth European Conference on Pattern Languages of Programs, Irsee, Germany, July 6-10*, volume 81, pages 431–470.

[3] Babuji, Y., Woodard, A., Li, Z., Katz, D. S., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J. M., Foster, I., Wilde, M., and Chard, K. (2019). Parsl: Pervasive parallel programming in python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing, June*, HPDC '19, page 25–36, New York, NY, USA. Association for Computing Machinery.

[4] Barker, A. and Hemert, J. V. (2008). Scientific workflow: A survey and research directions. *Parallel Processing and Applied Mathematics Lecture Notes in Computer Science, vol 4967. September 9-12. Springer, Berlin, Heidelberg.*, 4967:746–753.

[5] Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. In *IEEE Cloud Computing, September*, volume 1, pages 81–84. IEEE.

[6] Bertolino, A., Calabrò, A., Lonetti, F., and Sabetta, A. (2011). GLIMPSE: A Generic and Flexible Monitoring Infrastructure. In *13th European Workshop on Dependable Computing (EWDC), May*, pages 73–78, Pisa, Italy. ACM.

[7] Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. *IEEE Software, Mar.-Apr.*, 32(2):50–54.

[8] Chen, L. (2017). Continuous delivery: Overcoming adoption challenges. *Journal of Systems and Software, February*, 128:72 – 86.

[9] Chittaro, L., Guida, G., Tasso, C., and Toppano, E. (1993). Functional and teleological knowledge in the multimodeling approach for reasoning about physical systems: a case study in diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics, Nov.*, 23(6):1718–1751.

[10] Combacau, M., Berruet, P., Zamai, E., Charbonnaud, P., and Khatab, A. (2000). Supervision and monitoring of production systems. *IFAC Proceedings Volumes*, 33(17):849–854. 2nd IFAC Conference on Management and Control of Production and Logistics (MCPL 2000), Grenoble, France, 5-8 July.

[11] Datadog (2020). Cloud Monitoring as a Service. `https://www.datadoghq.com/`. Accedido: 01-14-2020.

[12] Deelman, E., Mehta, G., Singh, G., Su, M.-H., and Vahi, K. (2007). Pegasus: Mapping large-scale workflows to distributed resources. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science: Scientific Workflows for Grids*, pages 376–394, London. Springer London. January.

[13] Di Francesco, P., Malavolta, I., and Lago, P. (2017). Research on architecting microservices: trends, focus, and potential for industrial adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 21–30. IEEE. April.

[14] Docker (2020a). Docker stats. `https://docs.docker.com/engine/reference/commandline/stats/`. Accedido: 01-04-2020.

[15] Docker (2020b). Enterprise Container Platform | Docker. https://www.docker.com/. Accedido: 01-01-2020.

[16] Dragoni, N., Giallorenzo, S., Lluch-Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., and Safina, L. (2016). Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering*, volume abs/1606.04036. 13 Jun.

[17] Dreher, M. and Peterka, T. (2017). Decaf: Decoupled dataflows for in situ high-performance workflows. volume 1. Argonne National Laboratory (ANL), Argonne, Illinois, USA. July 31.

[18] Dynatrace (2020a). Docker monitoring | Dynatrace. https://www.dynatrace.com/technologies/docker-monitoring/. Accedido: 01-15-2020.

[19] Dynatrace (2020b). Software Intelligence for the Enterprise Cloud | Dynatrace. https://www.dynatrace.com. Accedido: 01-16-2020.

[20] Fahringer, T., Prodan, R., Duan, R., Hofer, J., Nadeem, F., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.-L., Villazon, A., and Wieczorek, M. (2007). ASKALON: A Development and Grid Computing Environment for Scientific Workflows. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science: Scientific Workflows for Grids*, pages 450–471, London. Springer London. January.

[21] Fatema, K., Emeakaroha, V. C., Healy, P. D., Morrison, J. P., and Lynn, T. (2014). A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. volume 74, pages 2918–2933. Elsevier. October.

[22] Foundation, C. N. C. (2020). Prometheus - Monitoring system time series database. https://prometheus.io.

[23] FP7-ICT (2020). Specific Programme "Cooperation": Information and communication

technologies | Programme | FP7 | CORDIS | European Commission. https://cordis.europa.eu/programme/id/FP7-ICT. Accedido: 01-26-2020.

[24] Gonzalez-Compean, J., Sosa-Sosa, V., Diaz-Perez, A., Carretero, J., and Yanez-Sierra, J. (2018a). Sacbe: A building block approach for constructing efficient and flexible end-to-end cloud storage. volume 135, pages 143 – 156. January.

[25] Gonzalez-Compean, J., Sosa-Sosa, V. J., Diaz-Perez, A., Carretero, J., and Marcelin-Jimenez, R. (2018b). Fedids: a federated cloud storage architecture and satellite image delivery service for building dependable geospatial platforms. In *International journal of digital earth*, volume 11, pages 730–751. Taylor & Francis. 20 Jul.

[26] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. (1997). Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *Data mining and knowledge discovery*, volume 1, pages 29–53. Springer. March 01.

[27] Heger, C., van Hoorn, A., Mann, M., and Okanović, D. (2017). Application performance management: State of the art and challenges for the future. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, pages 429–432.

[28] Hilman, M. H., Rodriguez, M. A., and Buyya, R. (2020). Multiple workflows scheduling in multi-tenant distributed systems: A taxonomy and future directions. volume 53, New York, NY, USA. Association for Computing Machinery. Feb.

[29] Hua, Y., Liu, X., and Jiang, H. (2014). Antelope: A semantic-aware data cube scheme for cloud data center networks. *IEEE Transactions on Computers*, 63(9):2146–2159.

[30] Humble, J. and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through*

Build, Test, and Deployment Automation. Addison-Wesley Signature Series (Fowler). Pearson Education.

[31] Imamagic, E. and Dobrenic, D. (2007). Grid infrastructure monitoring system based on nagios. In *Proceedings of the 2007 Workshop on Grid Monitoring*, GMW '07, pages 23–28, New York, NY, USA. ACM.

[32] Keutzer, K. (1987). Dagon: Technology binding and local optimization by dag matching. In *24th ACM/IEEE Design Automation Conference*, pages 341–347. June.

[33] Li, Q. and Chen, Y.-L. (2009a). Data flow diagram. In *Modeling and Analysis of Enterprise and Information Systems*, pages 85–97. Springer, Berlin, Heidelberg.

[34] Li, Q. and Chen, Y.-L. (2009b). *Modeling and Analysis of Enterprise and Information Systems: from requirements to realization*. Springer.

[35] Luko, S. (2013). Risk management principles and guidelines. *Quality Engineering*, 25.

[36] MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., and Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, 12(S 18).

[37] Massie, M. L., Chun, B. N., and Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817 – 840.

[38] Microsoft (2020). Cloud Computing Services | Microsoft Azure. `https://azure.microsoft.com/en-us/`. Accedido: 01-20-2020.

[39] Morabito, R., Kjällman, J., and Komu, M. (2015). Hypervisors vs. lightweight virtualization: a performance comparison. In *2015 IEEE International Conference on Cloud Engineering*, pages 386–393. IEEE.

[40] Networks, J. (2020). AppFormix Overview - TechLibrary - Juniper Networks. https://www.juniper.net/documentation/en_US/appformix/topics/concept/about-appformix.html. Accedido: 01-17-2020.

[41] Openstack.org (2020). Build the future of Open Infrastructure. https://www.openstack.org/.

[42] Ortega-Arjona, J. L. (2010). *Patterns for parallel software design*, volume 21. John Wiley & Sons.

[43] Osis, J. and Donins, U. (2017). Chapter 2 - software designing with unified modeling language driven approaches. In Osis, J. and Donins, U., editors, *Topological UML Modeling*, Computer Science Reviews and Trends, pages 53 – 82. Elsevier, Boston.

[44] Palmer, N. (2009). Workflow management coalition. In LIU, L. and ÖZSU, M. T., editors, *Encyclopedia of Database Systems*, pages 3550–3550, Boston, MA. Springer US.

[45] Project, I.-A. (2020). Internet of things architecture | iot-a project | fp7 | cordis | european commission. https://cordis.europa.eu/project/id/257521. Accedido: 01-24-2020.

[46] Rabin, M. O. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM (JACM)*, 36(2):335–348.

[47] Rellermeyer, J. S., Duller, M., Gilmer, K., Maragkos, D., Papageorgiou, D., and Alonso, G. (2008). The software fabric for the internet of things. In Floerkemeier, C., Langheinrich, M., Fleisch, E., Mattern, F., and Sarma, S. E., editors, *The Internet of Things*, pages 87–104, Berlin, Heidelberg. Springer Berlin Heidelberg.

[48] Reyes-Anastacio, H. G., Gonzalez-Compean, J., Sosa-Sosa, V. J., Carretero, J., and Blas, J. F. G. (2020). Kulla, a container-centric construction model for building infrastructure-agnostic distributed and parallel applications. *Journal of Systems and Software*, page 110665.

[49] Robinson, S., Arbez, G., Birta, L. G., Tolk, A., and Wagner, G. (2015). Conceptual modeling: Definition, purpose and benefits. In *2015 Winter Simulation Conference (WSC)*, pages 2812–2826.

[50] Siu, S. and Singh, A. (1997). Design patterns for parallel computing using a network of processors. In *Proceedings. The Sixth IEEE International Symposium on High Performance Distributed Computing (Cat. No. 97TB100183)*, pages 293–304. IEEE.

[51] Sousa, T. B. (2012). Dataflow programming concept, languages and applications. In *Doctoral Symposium on Informatics Engineering*, volume 130.

[52] Sysdig (2020). Secure DevOps Platform for Cloud-Native | Sysdig. `https://sysdig.com/`. Accedido: 01-13-2020.

[53] Vassiliadis, P., Simitsis, A., and Skiadopoulos, S. (2002). Conceptual modeling for etl processes. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 14–21. ACM.

[54] Vazquez-Martinez, G. A., Gonzalez-Compean, J., Sosa-Sosa, V. J., Morales-Sandoval, M., and Perez, J. C. (2018). Cloudchain: A novel distribution model for digital products based on supply chain principles. *International Journal of Information Management*, 39:90 – 103.

[55] W3C, Kaebisch, S., Kamiya, T., McCool, M., Charpenay, V., and Kovatsch, M. (2020). Web of Things (WoT) Thing Description. `https://www.w3.org/TR/wot-thing-description/`.

[56] Zeng, D., Guo, S., and Cheng, Z. (2011). The web of things: A survey. *JCM*, 6(6):424–438.