PerfSPEC Learning Phase

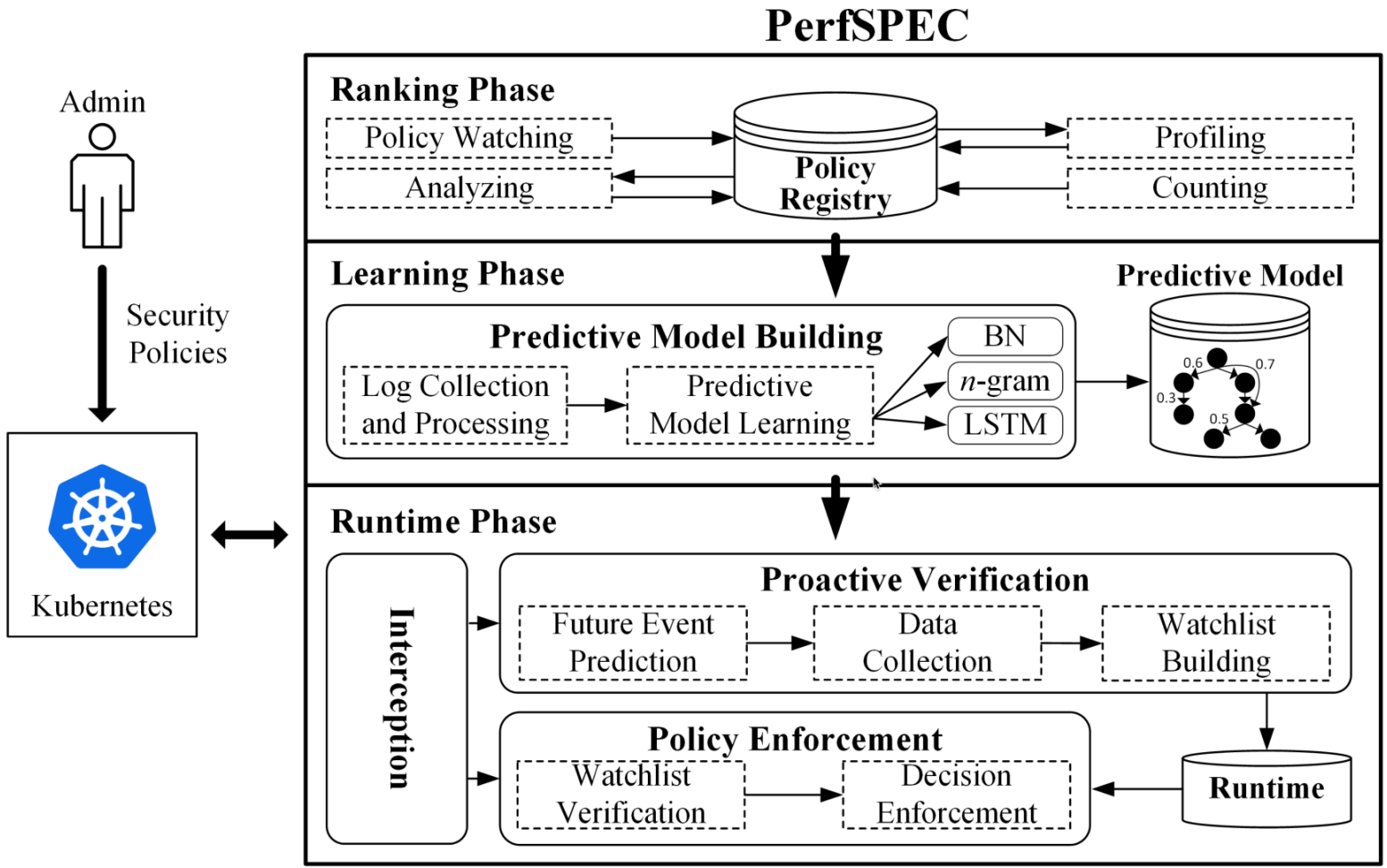
Aplicación proactiva de políticas de seguridad
basadas en perfiles de rendimiento
para contenedores

H. Kermabon-Bobinnec et al.,
"PerfSPEC: Performance Profiling-based Proactive Security Policy Enforcement for Containers,"
in IEEE Transactions on Dependable and Secure Computing,
doi: 10.1109/TDSC.2024.3420712

Aplicación proactiva de políticas de seguridad basadas en perfiles de rendimiento para contenedores

Performance Profiling-based Proactive Security Policy Enforcement for Containers -- IEEE publication

Abstract—Container environments provide cloud native applications with scalability, flexibility, and portable support. As a popular container orchestrator, Kubernetes facilitates automatic deployment and maintenance of a large number of containerized applications. However, potential misconfigurations, vulnerabilities, or implementation flaws may empower attackers to exploit the Kubernetes cluster. Although existing solutions such as runtime security policy enforcement may prevent an attack, they can be inefficient in large scale container environ- ments. In this paper, we propose a performance profiling-based proactive security policy enforcement solution, namely, PerfSPEC. First, we accelerate the proactivization of policies (which typically requires significant manual effort) by proposing to profile and rank existing policies according to their induced overhead. This allows us to better focus our efforts and greatly improve the overall response time (e.g., by 98% in contrast to less than 49%). Then, we address the performance limitations of existing solutions by leveraging learning-based approaches to predict future events and compute their verification results in advance. As a result, PerfSPEC achieves a viable response time (e.g., less than 10 ms in contrast to 600 ms with one of the most popular existing approaches) even for large container environments (up to 800 Pods).



PerfSPEC Learning Phase

Proceso

- ### Recolección logs

  Recopilación de logs de eventos del entorno. Para ello, PerfSPEC primero requiere habilitar la función de **auditoría en logs** de Kubernetes

- ### Procesado

  - Se extraen los eventos de los datos históricos y se genera un log de autitoría realista y representativo.
  - Se extraen los campos *objectRef[resource]* y métodos de los logs de eventos y se almacenan en un archivo
  - Usando herramientas de análisis de datos, se procesa cada entrada tipificando los eventos, a los que se asigna al par (método, recurso) una cadena *metodo_recurso* (tipo de evento).

- ### Modelo Aprendizaje Predictivo

  Implementar una de las tres diferentes aproximaciones propuestas: *Bayesian network*, *n-gram* y *LSTM*.

Recolección logs

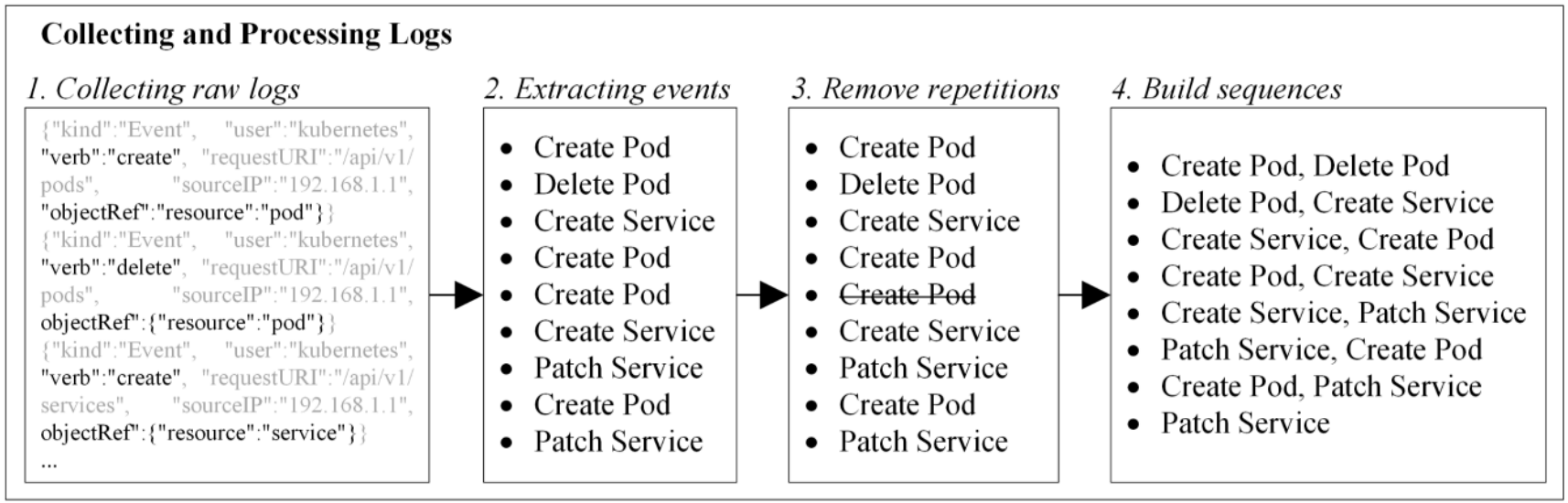
Metodología

- Para generar eventos, se adoptan los 60 **Helm** más populares disponibles en ArtifactHub que se implementan y eliminan a intervalos regulares para simular operaciones administrativas.
- Como estos paquetes de software son utilizados comúnmente por los usuarios para implementar aplicaciones y servicios en Kubernetes, se considera que las secuencias de operaciones realizadas para cada implementación y eliminación son **realistas y representativas**.
- Guardamos los eventos utilizando los logs de auditoría de Kubernetes

DataSet

- 60 paquetes
- 16.548 eventos
- 94.287 logs
- 6.489 acciones

Procesado



Preprocesamiento de datos: recopilación para generar información limpia y útil (acciones críticas) para entrenar modelos y clasificar prepare_perfspec.py
=> HTML prepare_perfspec.html |
=> Public HTML prepare_perfspec.html |

Aprendizaje Predictivo

Approach	Size of Window	Accuracy	Offline Learning Time	Runtime Inference Time
Bayesian Network	*N.A.*	79.7%	4.29 s	1e-4 s
LSTM	1	92.3%	24.01 s	0.06 s
	2	97.6%	32.75 s	0.07 s
n-grams	1	88.2%	0.01 s	1e-4 s
	2	97.3%	0.07 s	2e-4 s

LSTM

Long Short Term Memory

- Accuracy metric with loss
- Recall metric validation only
- F1 score metric validation only
- Precission metric validation only
- Checkpoints models
- Early Stopping
- Early ReduceLROnPlateu

- **Entrenar modelos** para obtener predicciones train_perfspec.py.
- **Obtener predicciones** de modelos existentes run_perfspec.py.
- **Revisión y análisis** de modelos entrenados (mejorar resultados) model_perfspec.py

  => HTML model_perfspec.html – Public HTML model_perfspec.html.

Objetivos

- *Look & feel*, interacciones entre procesamiento, análisis y presentación.
- Gestión paquetes de software como uv para complementar **pip** Python.
- Notebook abierto compatible como Marimo, motores alternativos: Polars y Pandas.
- Configuraciones y estructuras para probar diferentes ajustes y opciones.
- Implementar un modelo personalizado de LSTM en notebooks.
- Diferentes métricas para aplicar a modelos de entrenamiento, ajustes personalizables y checkpoints.
- Notebooks como scripts de Python uso en línea de comandos: recopilar predicciones o entrenar modelos.
- Usar DRY: reutilizar código y centralizar configuraciones o cargar recursos (lib_perfspec).
- Dividir tareas en varios **notebooks** específicos.

PerfSPEC fue diseñado para funcionar
en la seguridad de la nube/contenedores,
uno de los pilares de la informática actual y futura,
donde los sistemas distribuidos prometen
una resiliencia, escalabilidad y rendimiento sin precedentes.

PerfSPEC

Learning Phase

Rust Developer & Cloud Architect

python™

PerfSPEC Learning Phase

CONDA Conda package manager

pandas Pandas Data Analysis

marimo Python notebooks

jupyter Python notebooks

TensorFlow Machine learning

UV Python package and project manager Rust

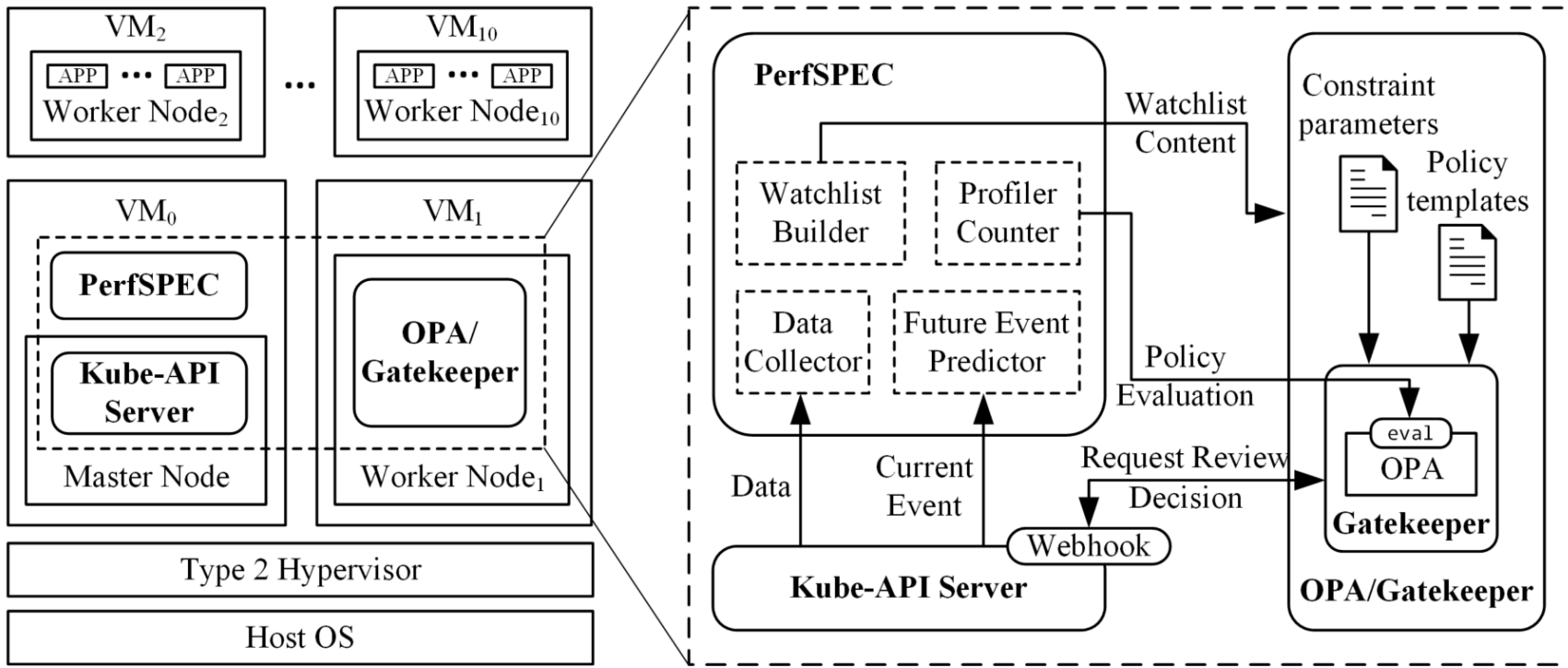
Polars DataFrames Rust

Nushell + Dataframes a Rust shell

Deep Learning Framework Rust

PerfSPEC Runtime Phase

Integración con Kubernetes



La seguridad tiene requisitos complejos,
es un ejercicio constante y sin fin
de mezclar y combinar múltiples elementos
que deben combinarse para crear
una forma de vida pacífica
como la de una orquesta y sus músicos
tocando frente a una audiencia.

PerfSPEC

Learning Phase

Rust Developer & Cloud Architect

Arquitectura inicial