

Deep Learning-Based Autoscaling Using Bidirectional Long Short-Term Memory for Kubernetes



Tabla de contenido

- Introducción y propuesta
- Diseño del sistema
- Autoscaler Personalizado Proactivo
- Modelo de Predicción Bi-LSTM
- Experimentación y Evaluación
- Resultados y Conclusiones

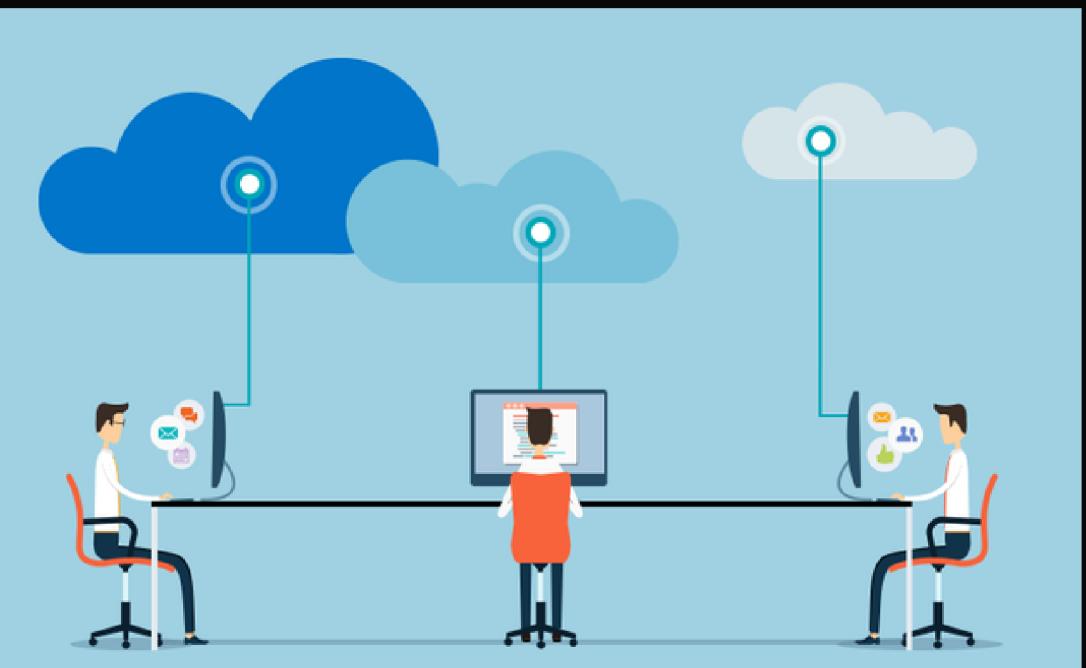
Introducción y propuesta

Cloud Computing

- Rápido desarrollo tecnológico en la última década.
- Popularidad creciente del cloud computing.

Elasticidad en Cloud Computing

- Gestión de cargas de trabajo impredecibles.
- Aprovisionar o desaprovisionar recursos según demanda.
- Objetivo: Mejorar rendimiento y reducir costos.



Desafíos del Autoscaling

- Riesgos de sobreaprovisionamiento y desaprovisionamiento.
- Dificultad en elegir valores de umbrales adecuados.
- Carga de trabajo fluctuante.

Enfoque Reactivo

- **Se basan en reglas con umbrales definidos por métricas de infraestructura.**
 - **Ejemplos: Utilización de CPU y memoria.**
- **Desafío principal: Elegir valores de umbrales adecuados.**
- **Causa del desafío: Fluctuaciones en la carga de trabajo debido al comportamiento variable del usuario.**

Enfoque Proactivo

- Popularidad del análisis de series temporales.
- Escasez de enfoques basados en algoritmos de ML.



En este paper se diseña un marco de autoscaling proactivo para Kubernetes

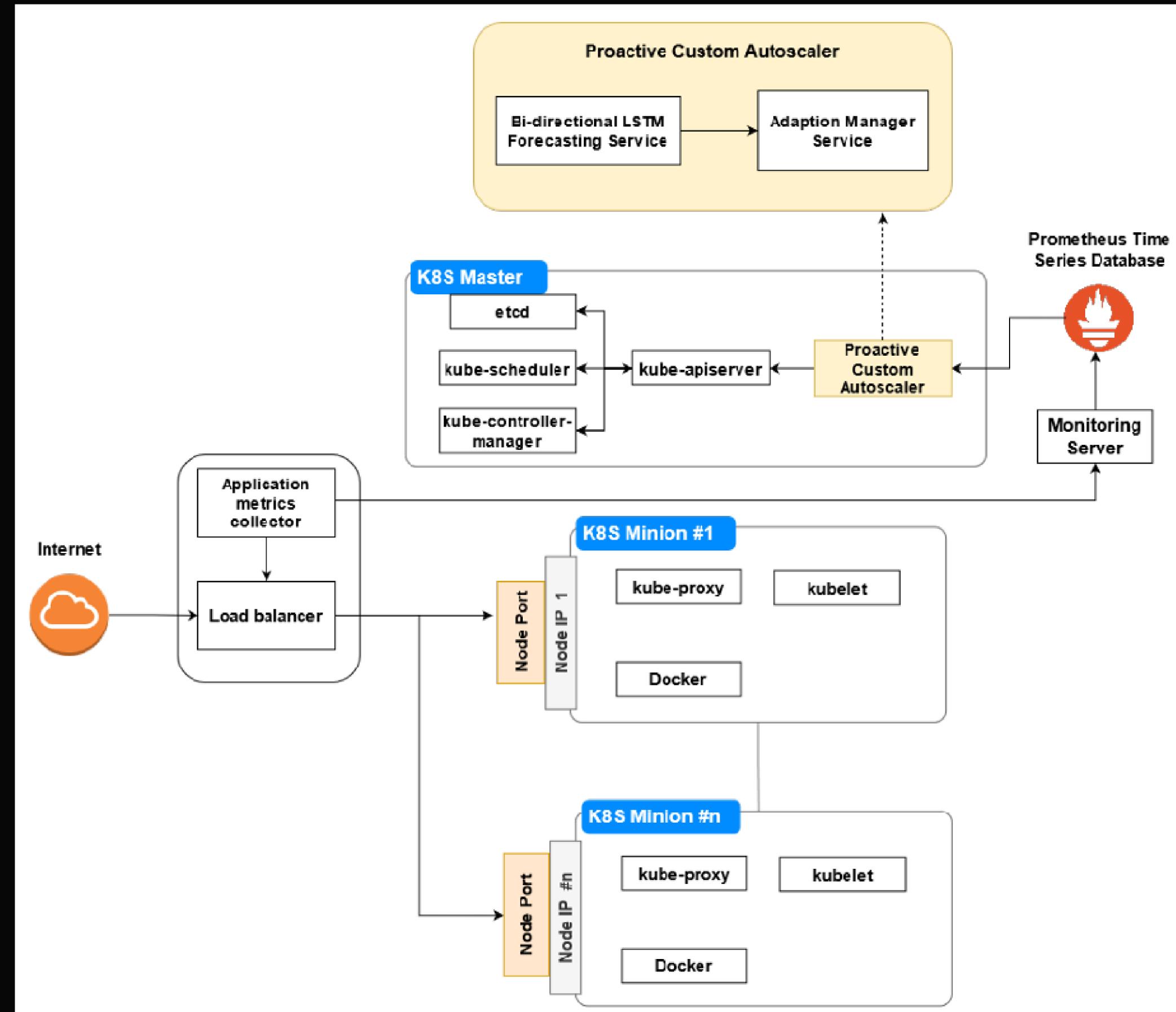
- 1) Se propone una arquitectura de sistema de autoscaling proactivo basada en Bi-LSTM diseñada para Kubernetes,
- 2) Se implementa el modelo de predicción Bi-LSTM para análisis de series temporales, mostrando que puede superar los modelos de predicción de autoscaling proactivo existentes.
- 3) Se propone una estrategia de eliminación de recursos (RRS) para eliminar parte del recurso cuando la carga de trabajo disminuye, así puede manejar mejor un pico de carga de trabajo que ocurra en el futuro cercano.

Diseño del sistema

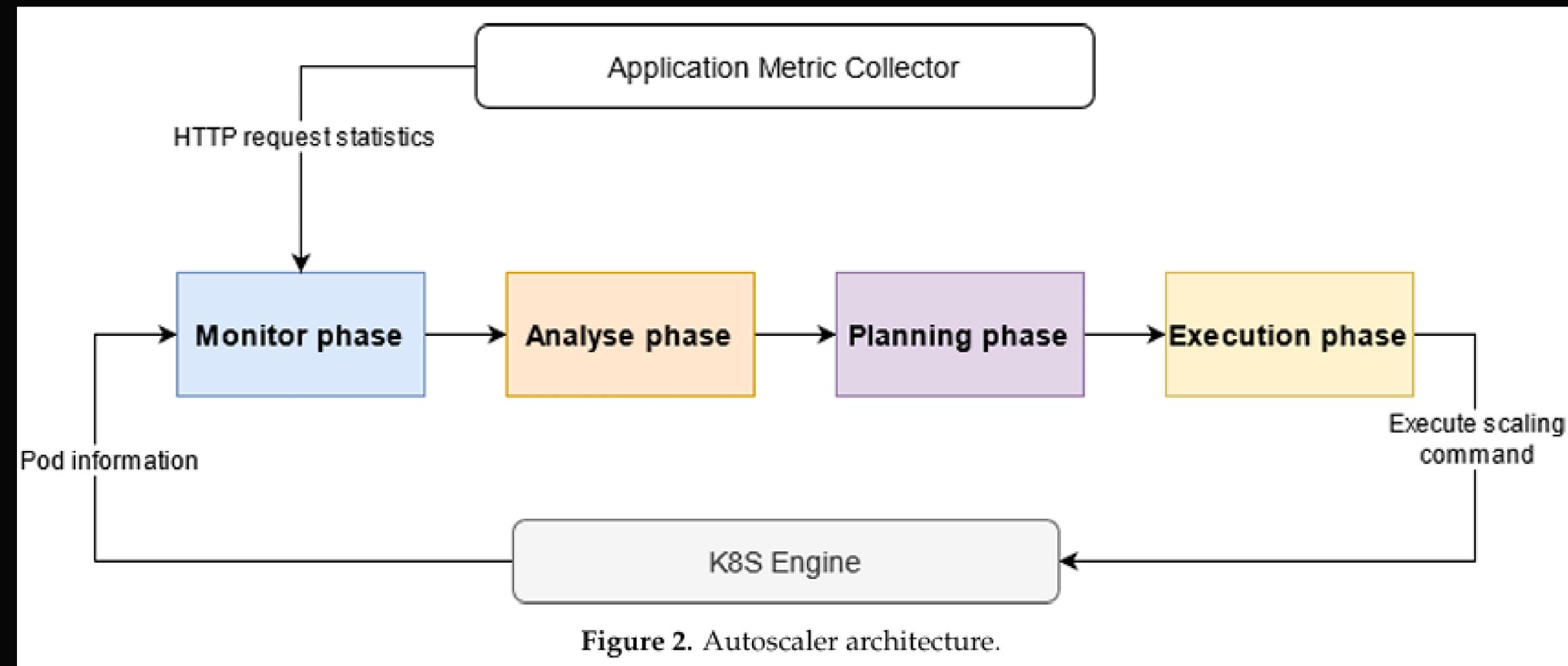
El diseño de la arquitectura del sistema, que se basa en el bucle monitor-analizar-planear-ejecutar (MAPE).

La arquitectura propuesta incluye los siguientes componentes:

- LoadBalancer;
- Recolector de métricas de aplicación;
- MonitorServer;
- K8SMaster: etcd, kube-scheduler, kube-control-manager, kuber-apiserver, Autoscaler Personalizado Proactivo;
- K8SMinions: kube-proxy, kubelet, Docker.



Autoscaler Personalizado Proactivo



Fase de Monitoreo

- Recepción continua de datos.
- Dos fuentes principales: Datos de red y información del pod.
- Agregación y envío a Prometheus.

Fase de Análisis

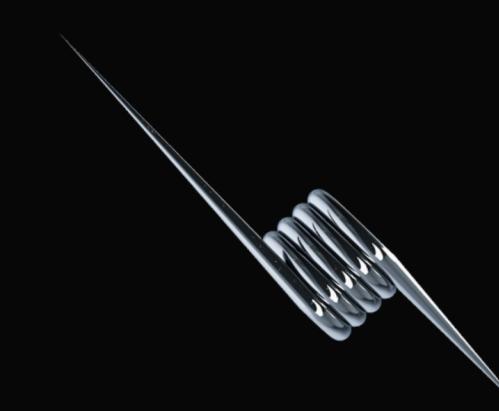
- Uso de la API restful de Prometheus.
- Modelo de red neuronal Bi-LSTM.
- Funcionamiento bidireccional de Bi-LSTM.

Fase de Planificación

- Cálculo de pods requeridos.
- Estrategia de escalado basada en el Algoritmo 1.
- Estrategia de eliminación de recursos (RRS).

Fase de Ejecución

- Acción final del bucle MAPE.
- Motor Kubernetes (kubernetesapiserver).
- Cambio en el número de réplicas de pod.



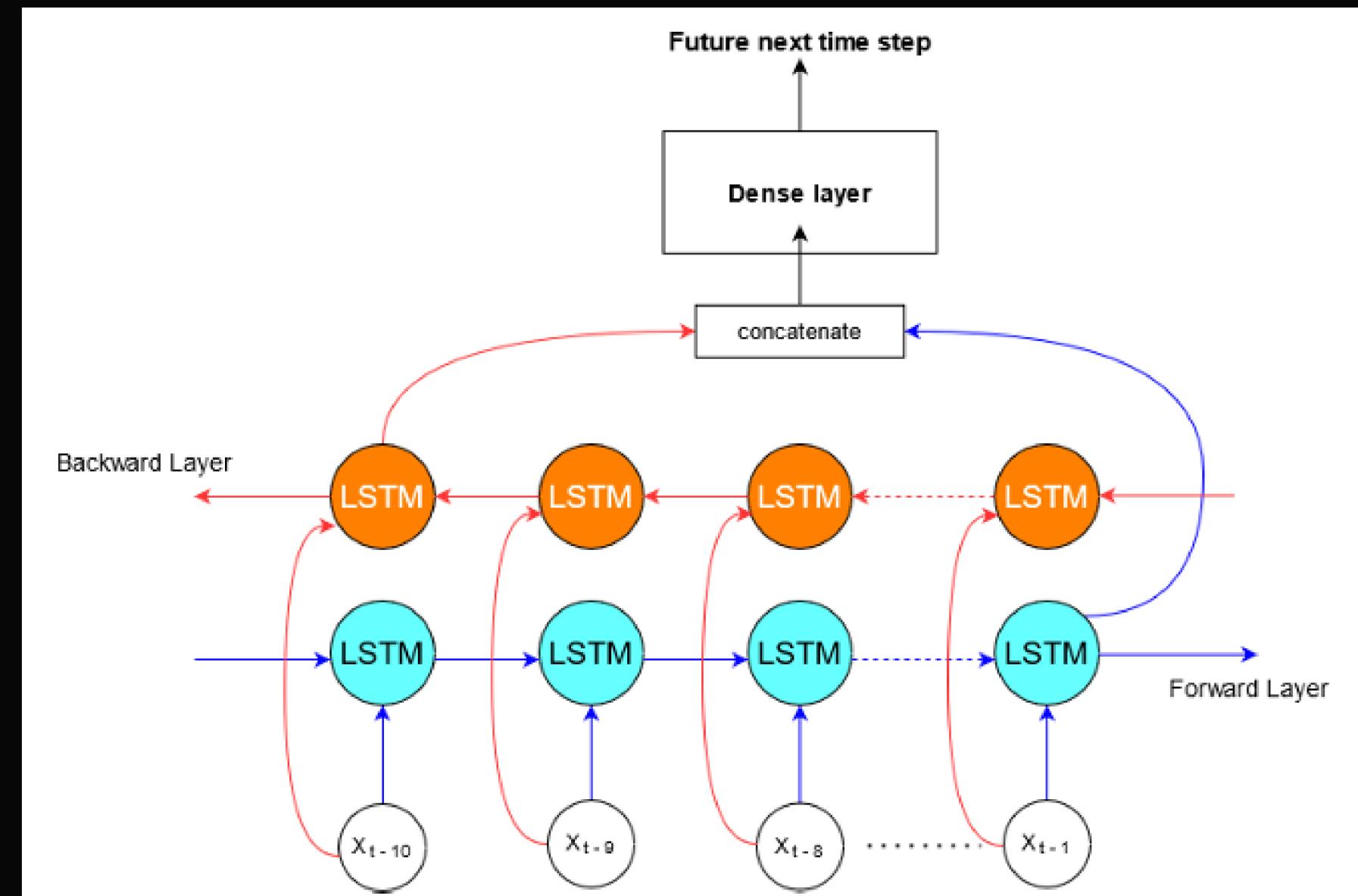
Fase de Planificación

- Cálculo de pods requeridos.
- Estrategia de escalado basada en el Algoritmo 1.
- Estrategia de eliminación de recursos (RRS).

Algorithm 1: Adaption Manager Service algorithm.

```
Data:  $Workload_{t+1}$  //predicted workload in the next interval  
Result: Scaling action  
initialization;  
while system is running do  
  for each CDT do  
     $pods_{t+1} = \frac{Workload_{t+1}}{Workload_{pod}}$  ;  
    if  $pods_{t+1} > pods_t$  then  
      EXECUTE_SCALE_OUT_COMMAND ( $pods_{t+1}$ ) ;  
    else if  $pods_{t+1} < pods_t$  then  
       $pods_{t+1} = \max(pods_{t+1}, pods_{min})$  ;  
       $pods_{surplus} = (pods_t - pods_{t+1}) * RRS$  ;  
       $pods_{t+1} = pods_t - (pods_{surplus})$  ;  
      EXECUTE_SCALE_IN_COMMAND ( $pods_{t+1}$ ) ;  
    else  
      Do nothing ;  
    end  
  end  
end
```

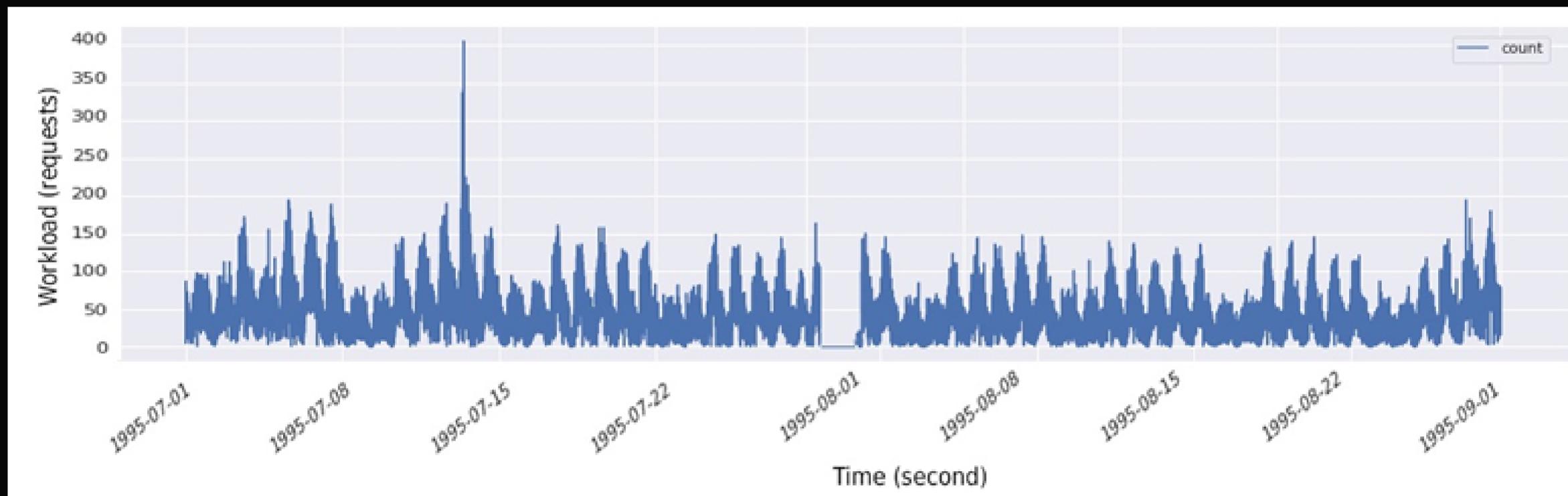
Modelo de Predicción Bi-LSTM



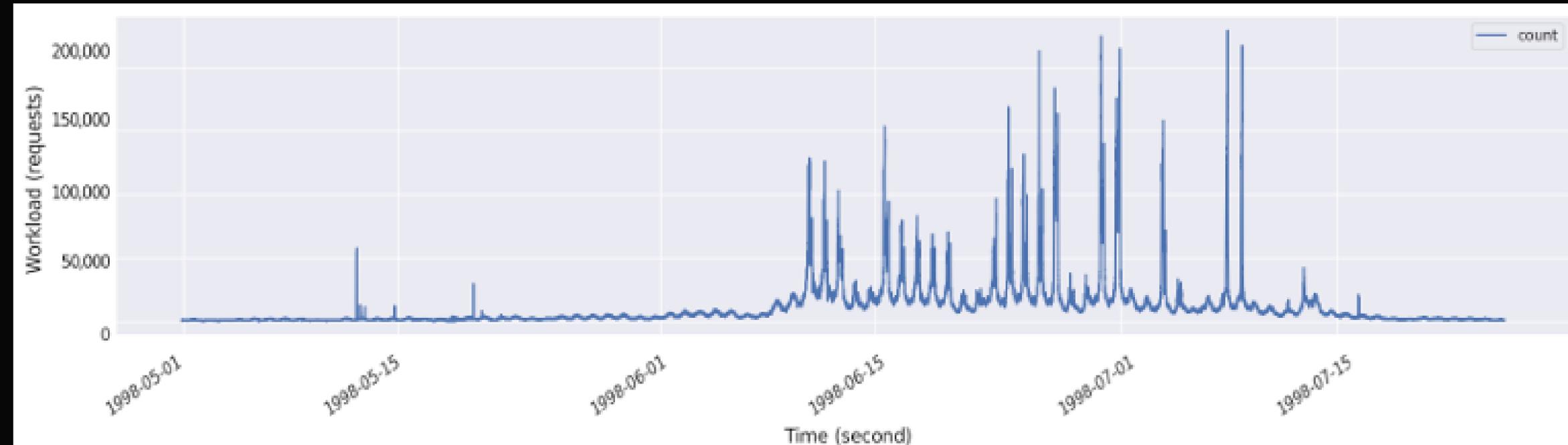
Experimentación y Evaluación

Primero, se evalúa el modelo Bi-LSTM propuesto y su precisión y velocidad de predicción se comparan con las del LSTM y ARIMA en utilizando diferentes conjuntos de datos. Posteriormente, se simula y evalúa un sistema propuesto completo en un entorno real para ver el rendimiento en términos de precisión de aprovisionamiento y aceleración elástica.

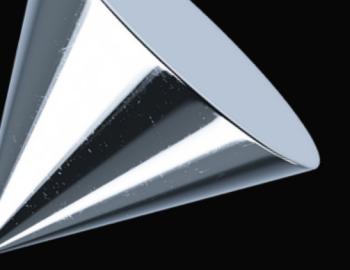
Datasets



Dos meses de servidores web NASA [27].



Tres meses de servidores web FIFA World Cup 98.



Primer Experimento

- Uso de conjuntos de datos NASA y FIFA World Cup.
- División: 70% entrenamiento, 30% prueba.
- Comparación: Bi-LSTM vs. LSTM hacia adelante y ARIMA.

Segundo Experimento

- Simulación del entorno.
- Uso de un subconjunto de la base de datos FIFA World Cup 98.
- Assumptions: Un pod = un contenedor Docker, 500 peticiones HTTP/pod.

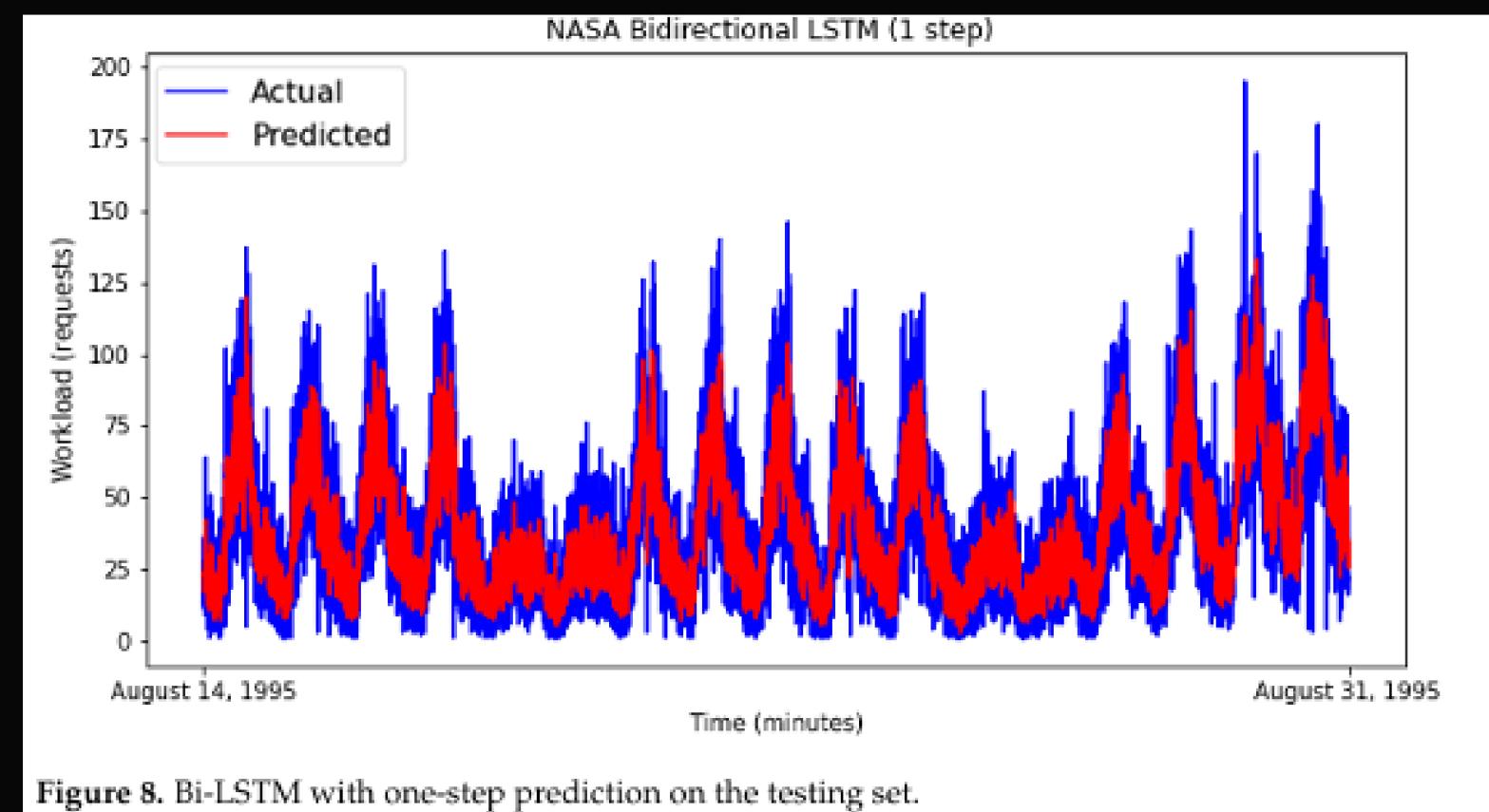
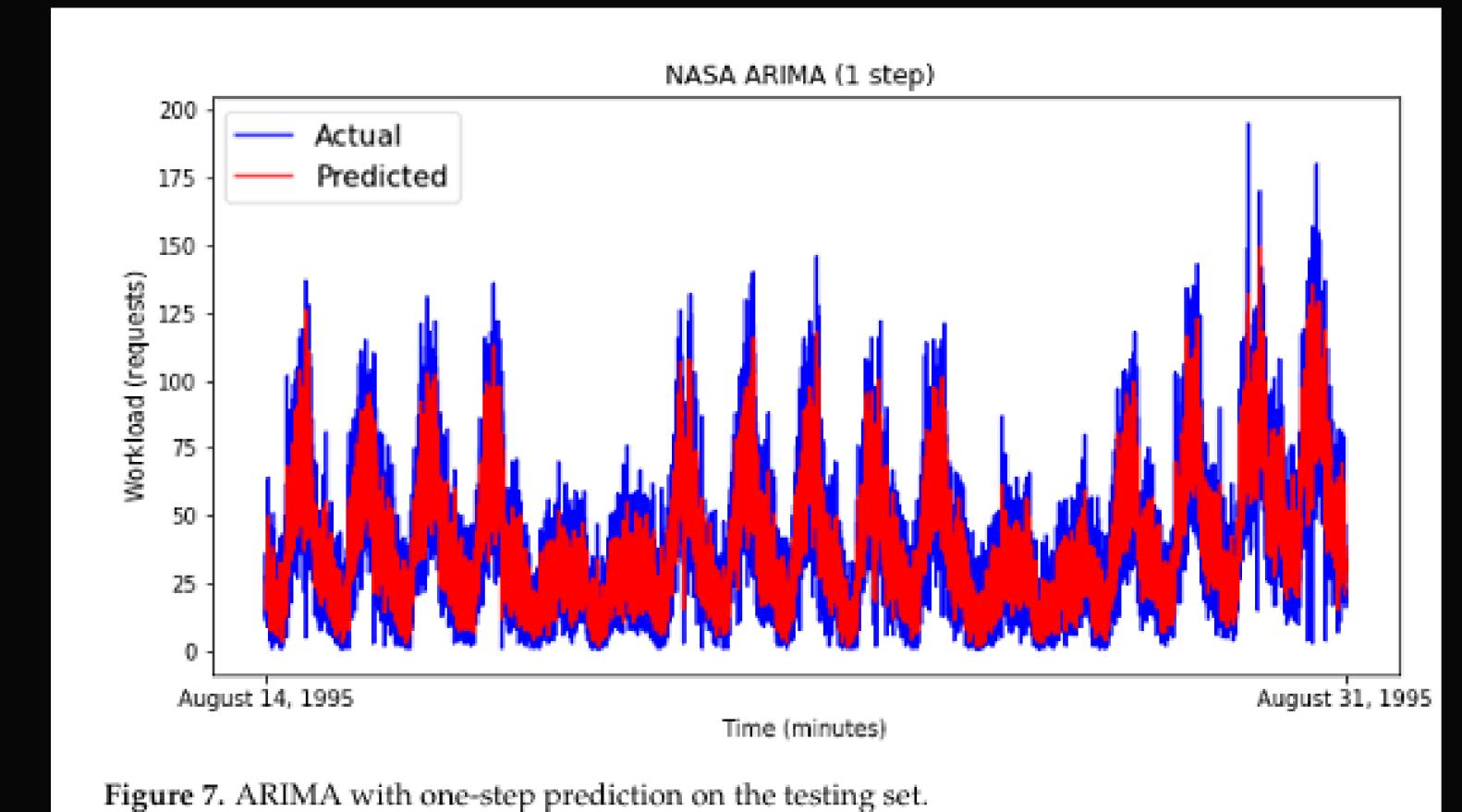
Resultados y Conclusiones

Conjunto de Datos NASA

- Modelo Bi-LSTM vs. ARIMA
- Métricas clave: MSE, RMSE, MAE
- Bi-LSTM tiene una velocidad de predicción 530 y 55 veces más rápida.
- ARIMA tiene mayores valores de error absoluto.
- Bi-LSTM superior en precisión y velocidad de predicción.

Model Type	ARIMA 1 Step	BI-LSTM 1 Step	ARIMA 5 Steps	BI-LSTM 5 Steps
MSE	196.288	183.642	237.604	207.313
RMSE	14.010	13.551	15.414	14.39
MAE	10.572	10.280	11.628	10.592
R^2	0.692	0.712	0.628	0.675
Prediction speed (ms)	2300	4.3	2488	45.1

Bold values indicate the best results.

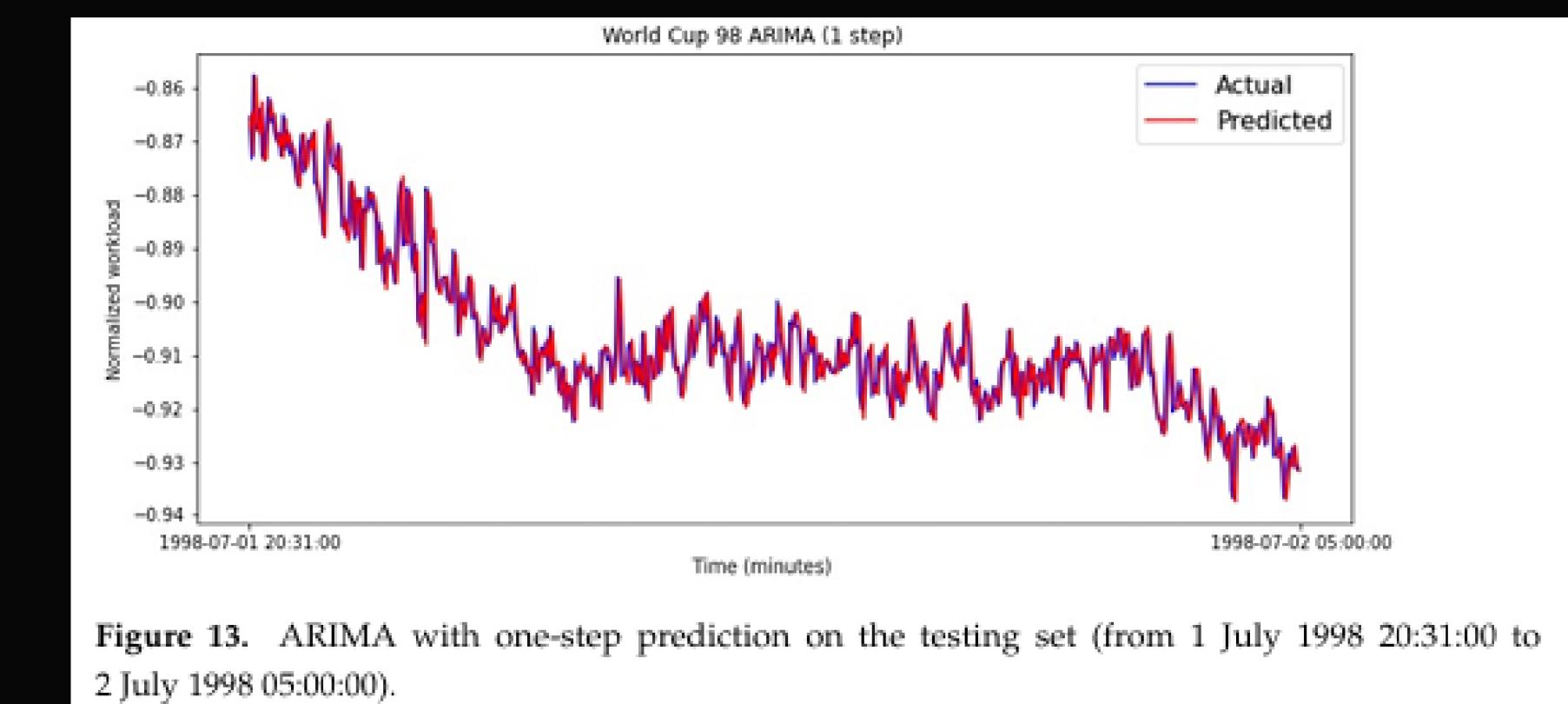
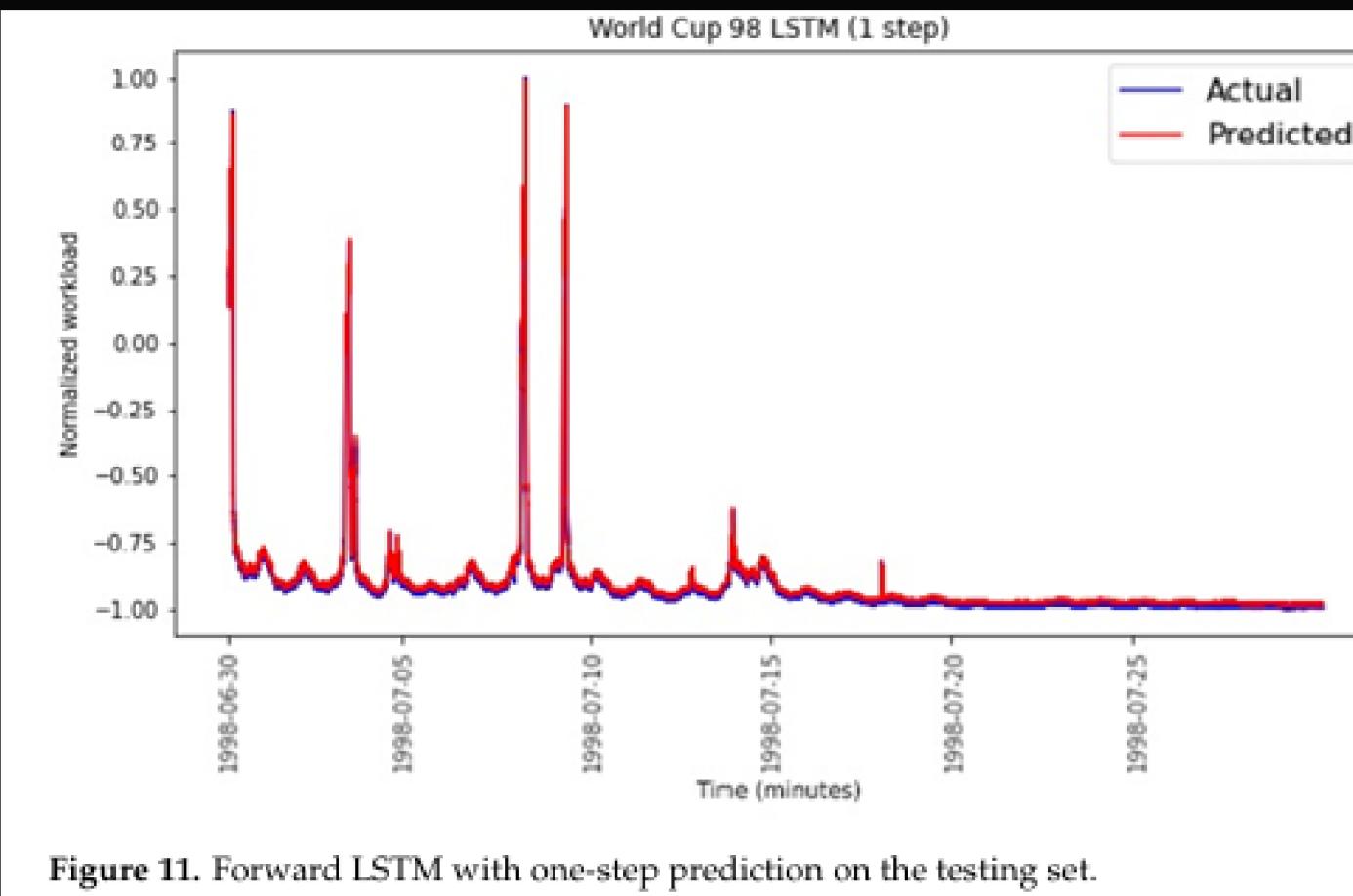
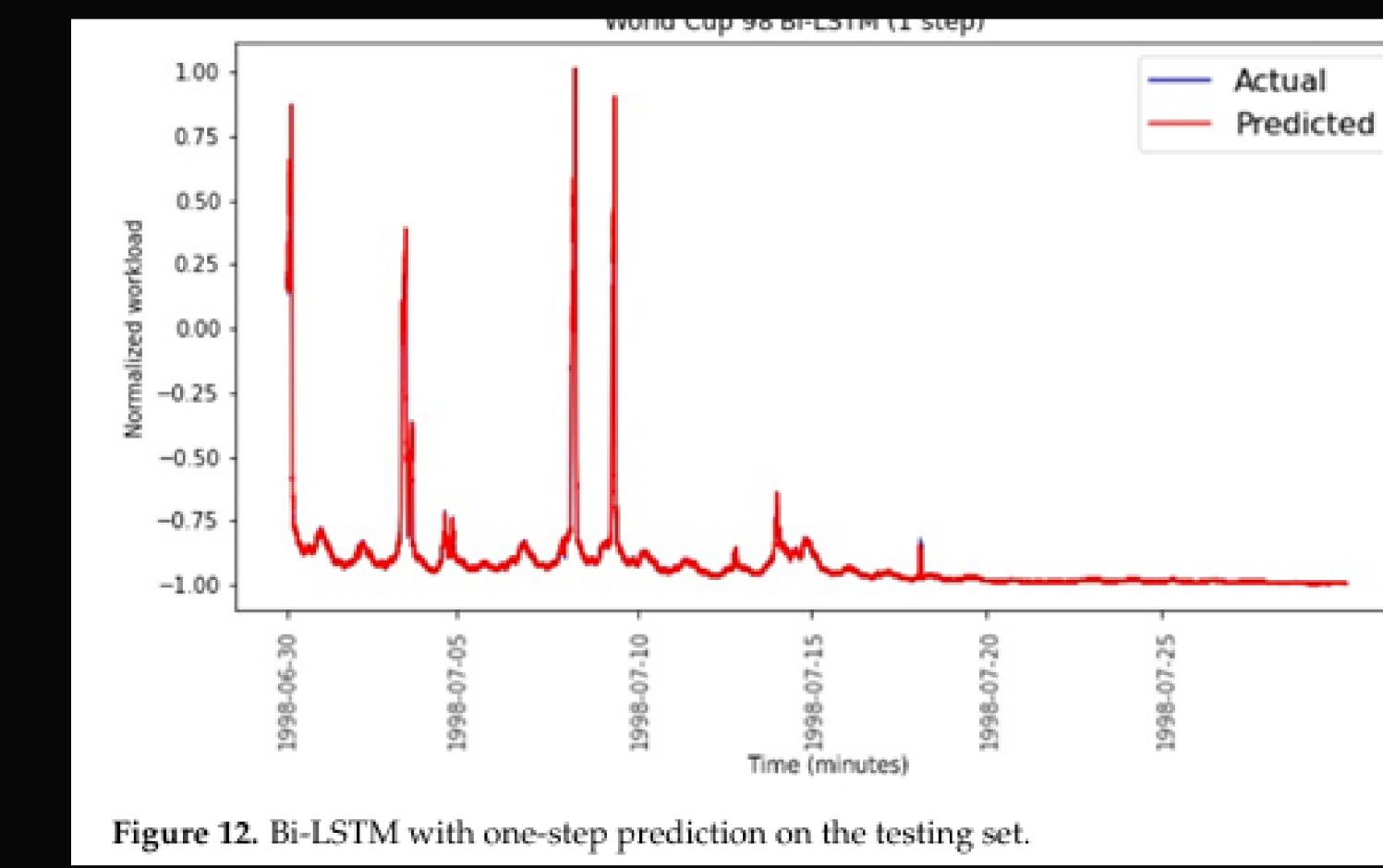
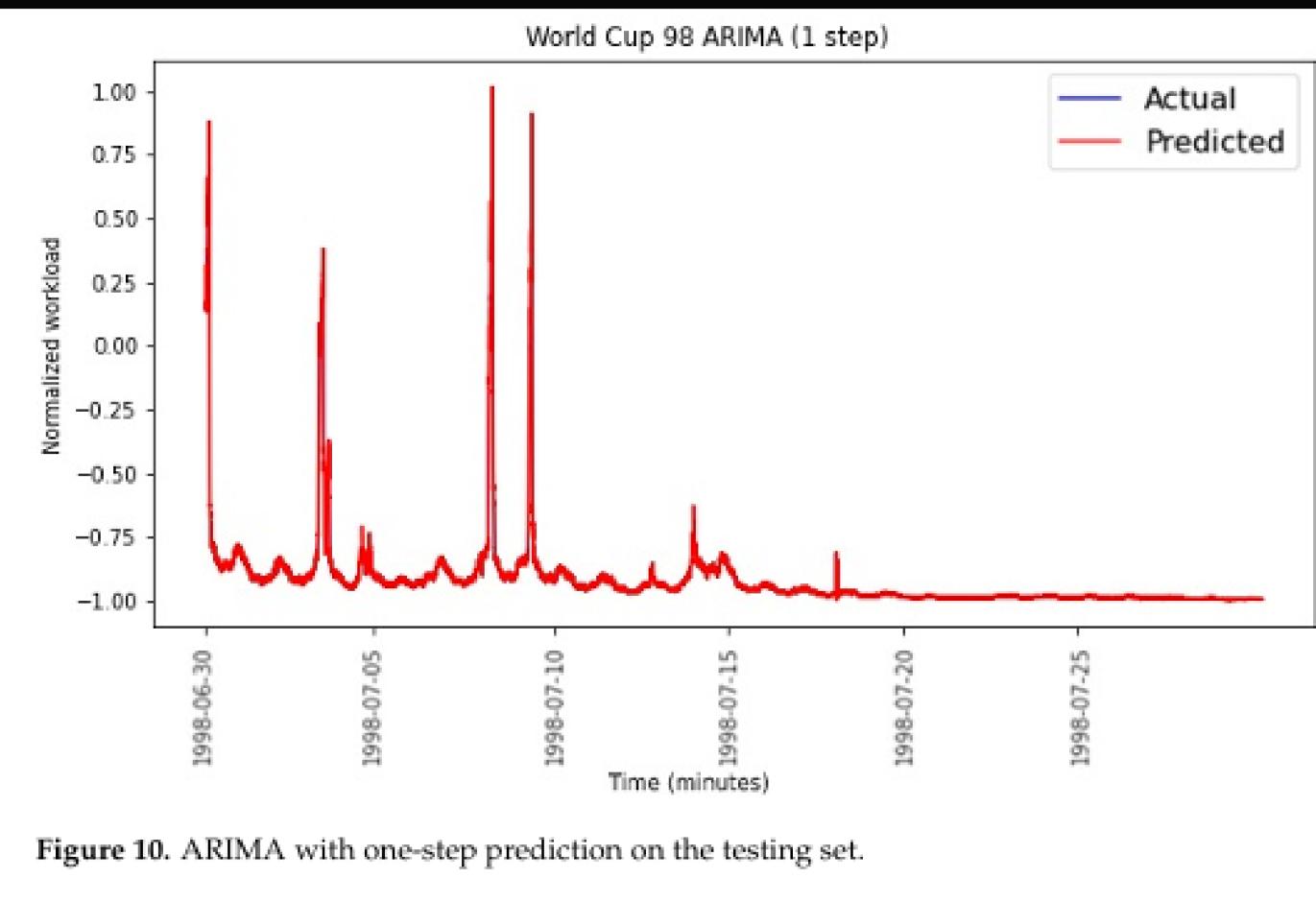


Conjunto de Datos FIFA World Cup 98

- Modelo Bi-LSTM vs. otros modelos.
- Superioridad en predicciones de un paso y cinco pasos.
- Velocidad de predicción 600 veces más rápida que ARIMA.

Model Type	ARIMA 1 Step	LSTM 1 Step	BI-LSTM 1 Step	ARIMA 5 Steps	LSTM 5 Steps	BI-LSTM 5 Steps
MSE	0.000040	0.000043	0.000036	0.000172	0.000157	0.000120
RMSE	0.006350	0.006523	0.006015	0.0131	0.0125	0.010
MAE	0.003302	0.003958	0.003127	0.0049	0.0068	0.00428
R^2	0.998496	0.998397	0.998637	0.9930	0.9941	0.9954
Prediction speed (ms)	3700	5.1	5.8	4076	49.6	51.2

Bold values indicate the best results.



Evaluación del Autoscaler

- ARIMA no es adecuado para autoscaling en tiempo real.
- Comparación entre Bi-LSTM y LSTM en términos de sobreaprovacionamiento y subaprovacionamiento.
- Bi-LSTM tiene una mayor ganancia de autoscaling sobre LSTM.

Table 6. Evaluation of auto-scaler.

Type	No-Autoscaling	LSTM 1 Step	BI-LSTM 1 Step	LSTM 5 Steps	BI-LSTM 5 Steps
Θ_O [%]	593.775	1.610	1.287	3.405	1.157
T_O [%]	97.60	49.400	42.80	55.6	18.2
Θ_U [%]	0.189	0.806	0.964	1.784	3.312
T_U [%]	2.2	30.0	29.8	35.80	63.2
ϵ_n	1	1.881	1.974	1.188	1.529

Bold values indicate the best results.

LSTM (1 step)

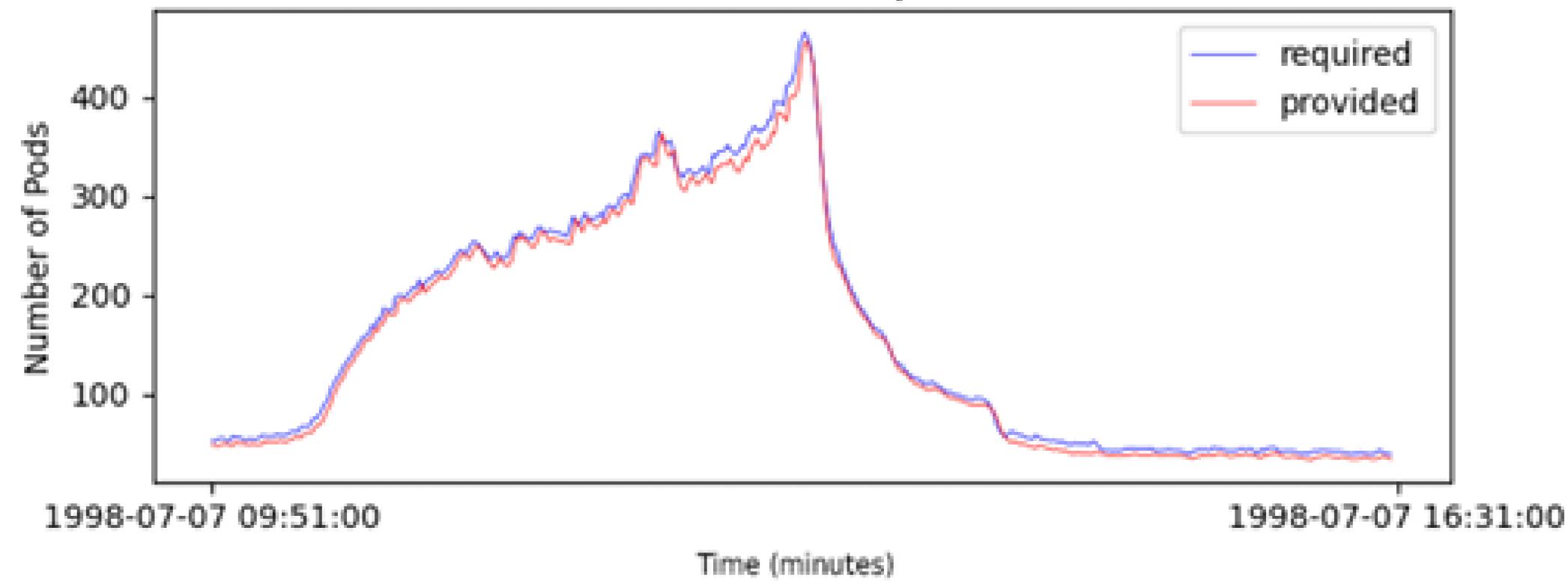


Figure 16. Number of pods produced by the LSTM auto-scaler.

Bi-LSTM (1 step)

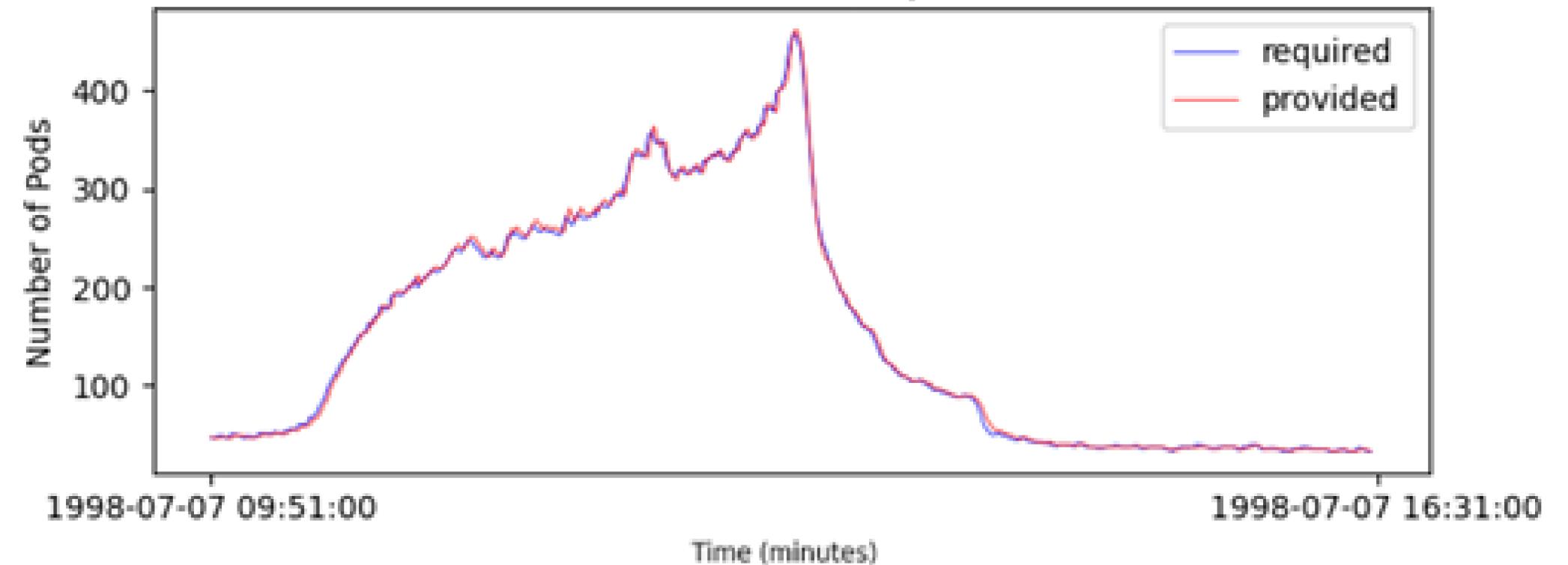


Figure 17. Number of pods produced by the Bi-LSTM auto-scaler.

Conclusiones

- Experimento realizado con diferentes conjuntos de datos.
- Comparación entre el modelo propuesto (Bi-LSTM), el modelo LSTM y el modelo estadístico ARIMA.
- El modelo propuesto supera en precisión a:
 - Modelo LSTM.
 - Modelo estadístico ARIMA.
- Eficacia en predicciones a corto y largo plazo.
- El modelo propuesto es de 530 a 600 veces más rápido que el ARIMA.
- Su velocidad es comparable a la del modelo LSTM, variando según las cargas de trabajo.
- El sistema propuesto vs. autoscaler horizontal de pods (HPA) de Kubernetes.
- Superioridad en:
 - Precisión.
 - Velocidad.
- Eficacia al aprovisionar y desaprovisionar recursos.
- Reafirmación de la eficacia del modelo propuesto (Bi-LSTM) en comparación con otros modelos y sistemas.

¡Gracias por su atención!

