Plan de Recuperación, Transformación y Resiliencia

Financiado por la Unión Europea NextGenerationEU

uc3m | Universidad Carlos III de Madrid

SORUS

Validación y optimización conjunta de RIS y vRANs

SORUS-RAN A2.1-E2

PERFILADO FINAL DE VRAN

Revisión	Autor	Fecha de entrega	Cambios
Versión 01	Jose Ayala Romero, Andrés García Saavedra	10/02/2024	Versión Inicial
Versión 02	Luis Roda Sánchez, Jorge San Martin Gomez	26/02/2024	Revisión, actualización de tablas, resumen ejecutivo y conclusiones

Exención de responsabilidad:

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

CONTENIDOS

LISTA DE ABREVIATURAS Y ACRÓNIMOS

Tabla 1. Lista de abreviaturas y acrónimos

Abreviatura	Explicación/Definición
BS	Base Station
CAT	Cache Allocation Technology
CP	Control Plane
DL	DownLink
FCAPS	Fault, Configuration, Accounting, Performance And Security
FDD	Frequency-Division Duplexing
FEC	Forward Error Correction
GPP	General Purpose Processor
IA	Inteligencia Artificial
IPC	Instructions Per Cycle
LLC	Last Level Cache
MAC	Media Access Control
MCS	Modulation and Coding Scheme
MPKI	Missed Predictions per 1000 (K) Instructions
NFV	Network Function Virtualization
O-CU	Open – Centralized Unit
O-DU	Open – Distributed Unit
OFDM	Orthogonal Frequency-Division Multiplexing
O-RU	Open – Radio Unit
PoE	Power-over-Ethernet
RAN	Radio Access Network
RIC	RAN Intelligent Controller
RRM	Radio Resource Management
RT	Real-Time

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

Abreviatura	Explicación/Definición
RU	Radio Unit
SMO	Service Management and Orchestration
SMT	Simultaneous MultiThreading
SNR	Signal-to-Noise Ratio
SO	Sistema Operativo
UE	User Equipment
UL	UpLink
UP	User Plane
vBS	virtualized Base Station
VNF	Virtualized Network Function
vRAN	virtualized Radio Access Network

LISTA DE FIGURAS

LISTA DE TABLAS

1 RESUMEN EJECUTIVO

En este documento se resumen las características existentes en entornos de vRAN y las soluciones a través de la arquitectura O-RAN propuesta, que incluye tanto el control inteligente de la radio en tiempo no real como casi real. Se aborda principalmente el problema de los *noisy neighbours* en este tipo de despliegues con compartición de recursos. Se presenta una plataforma experimental basada en Docker con la que se han realizado diferentes experimentos para encontrar la razón principal de la degradación exponencial del rendimiento con el aumento de vBS. Se ha llegado a la conclusión de que la computación segura afecta débilmente a dicho aumento (7%) y los cambios de contexto suponen un 43% de la sobrecarga total. Sin embargo, el principal problema reside en las pérdidas de caché, llegando a aumentar en un 500% con cinco vBS.

2 INTRODUCCIÓN

La virtualización de la red de acceso radioeléctrico (RAN) ha surgido como una tecnología fundamental en la optimización y evolución de los sistemas móviles de última generación. Su implementación no solo promete mejoras en la eficiencia operativa, sino también en la rentabilidad de estas redes. En un contexto donde la demanda de conectividad móvil se incrementa exponencialmente y donde la densidad de puntos de acceso radioeléctrico es cada vez mayor, surge la necesidad imperante de encontrar soluciones que puedan satisfacer estas demandas crecientes sin aumentar los costos asociados. Es en este escenario donde la virtualización de RAN (vRAN) emerge como una opción atractiva y prometedora.

Las iniciativas en este ámbito son numerosas y variadas. Desde la O-RAN Alliance[1] hasta el despliegue greenfield de Rakuten en Japón[2], la industria ha dirigido su atención hacia la virtualización de la RAN como una estrategia clave para alcanzar sus objetivos de eficiencia y rentabilidad. La centralización de las RAN no es una novedad en sí misma, pero su implementación en un contexto virtualizado promete llevar estas capacidades un paso más allá. De hecho, estudios recientes[3] indican que un considerable porcentaje de operadores en Estados Unidos tienen la intención de desplegar la centralización de RAN para el año 2025. Nombres como NTT Docomo, Ericsson o AT&T son ampliamente reconocidos por su interés y liderazgo en la adopción de estas tecnologías, lo que subraya aún más la relevancia y el impulso detrás de esta tendencia[4] [5] [6].

La promesa de la vRAN va más allá de la simple virtualización de recursos. Se espera que importe las ventajas ya conocidas de la Virtualización de Funciones de Red (NFV), permitiendo la compartición y multiplexación de recursos de infraestructura[7]. Sin embargo, aunque esta idea suena prometedora en teoría, todavía quedan preguntas por responder y desafíos por superar en la práctica. Uno de los aspectos menos estudiados hasta ahora es el impacto en tiempo real de la contención de recursos en plataformas compartidas de agrupación de RAN y cómo afectará esto a

[1] Garcia-Saavedra, A., & Costa-Perez, X. (2021). O-RAN: Disrupting the virtualized RAN ecosystem. *IEEE Communications Standards Magazine*, *5*(4), 96-103.

[2] Cisco, Rakuten, Altiostar, "Reimagining the End-to-End Mobile Network in the 5G Era," White Paper, 2019, Link.

[3] Heavy Reading, "5G Transport: A 2021 Heavy Reading Survey," White Paper, Feb. 2022.

[4] N. DOCOMO. (2013) Docomo to develop next-generation base stations utilizing advanced c-ran architecture for lte-advanced.

[5] Ericsson. (2021) Exploring new centralized ran and fronthaul opportunities.

[6] AT&T. (2022, Feb.) Cloudifying 5G with an Elastic RAN. [Online]. Available: HTTPS://ABOUT.ATT.COM/INNOVATIONBLOG/2022/CLOUDIFYING-5G-WITH-ELASTIC-RAN.HTML

[7] G. Garcia-Aviles et al., "Nuberu: Reliable RAN virtualization in shared platforms," in Proceedings of the 27th MobiCom, 2021, pp. 749–761.

la calidad del servicio y a la experiencia del usuario final[8]. Estas son cuestiones críticas que merecen una atención más detenida y una investigación más profunda.

El éxito previo de la NFV en la virtualización de funciones de red ha impulsado al mercado a desarrollar una variedad de Funciones de Red Virtuales (VNF), desde cortafuegos hasta conmutadores y VPNs, que ofrecen un rendimiento comparable al de operadores convencionales. Sin embargo, investigaciones han revelado que la contención de recursos causada por la compartición de infraestructura entre VNFs puede resultar en una degradación significativa del rendimiento, incluso hasta un 40% en comparación con plataformas dedicadas[9] [10]. Este fenómeno, conocido como "noisy neighbor", ha sido objeto de intensos estudios y ha motivado la búsqueda de soluciones efectivas para mitigarlo[11] [12] [13].

En el campo de la vRAN, se han realizado avances notables en los últimos años. En el ámbito industrial, soluciones como Intel FlexRAN y NVIDIA Aerial han demostrado su eficacia utilizando aceleradores de hardware especializados. Sin embargo, estos enfoques pueden enfrentar desafíos relacionados con su alto costo y consumo energético. Por otro lado, en el ámbito académico, proyectos como Agora[14] han explorado la viabilidad de ejecutar tareas de capa física (PHY) de RAN en plataformas de CPU multinúcleo de propósito general, siempre y cuando los recursos de CPU estén dedicados exclusivamente a tareas específicas. Más recientemente, Concordia[15] ha propuesto un enfoque innovador para compartir recursos de computación con aplicaciones de latencia elástica, aunque persisten interrogantes sobre la distribución óptima de recursos entre múltiples vBS.

[8] O-RAN Alliance, “Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN (O-RAN.WG6.CADS-v04.00) ,” Technical Report, Oct. 2022. [Sección 5.1.3]

[9] A. Tootoonchian et al., “ResQ: Enabling SLOs in Network Function Virtualization,” in Proceedings of the 15th USENIX NSDI, 2018, pp. 283–297.

[10] A. Manousis et al., “Contention-aware performance prediction for virtualized network functions,” in Proceedings of the ACM SIGCOMM, 2020, pp. 270–282.

[11] C. Sun et al., “NFP: Enabling network function parallelism in NFV,” in Proceedings of the ACM SIGCOMM, 2017, pp. 43–56.

[12] P. Kumar et al., “PicNIC: predictable virtualized NIC,” in Proceedings of the ACM SIGCOMM, 2019, pp. 351–366.

[13] J. Gong et al., “Microscope: Queue-based performance diagnosis for network functions,” in Proceedings of the ACM SIGCOMM, 2020, pp. 390–403.

[14] J. Ding et al., “Agora: Real-time massive MIMO baseband processing in software,” in Proceedings of the 16th CoNEXT, 2020, pp. 232–244.

[15] X. Foukas and B. Radunovic, “Concordia: Teaching the 5G vRAN to share compute,” in Proceedings of the ACM SIGCOMM, 2021, pp. 580–596.

A pesar de que el problema del "noisy neighbor" en cargas de trabajo NFV ha sido objeto de extensa investigación, se ha prestado poca atención al caso específico de la vRAN. Aunque algunas soluciones, como la proporcionada por Nuberu[16], han mejorado la fiabilidad del procesamiento RAN PHY frente a fluctuaciones de la capacidad de cómputo, la asignación eficiente de recursos de CPU sigue siendo un desafío pendiente. Por otro lado, vrAIn[17] ha abordado esta cuestión, pero sin considerar plenamente el impacto del "noisy neighbor" y sin soportar un número variable de vBS en el sistema. Por lo tanto, queda claro que aún hay mucho trabajo por hacer para abordar de manera efectiva y completa los desafíos asociados con la virtualización de la RAN en plataformas de computación compartida.

3 RELACIÓN CON ENTREGABLES PREVIOS Y ESTRUCTURA DEL DOCUMENTO

El entregable SORUS-RAN-A2.1-E1 desarrolla un conjunto de medidas experimentales relacionadas con el consumo energético y el rendimiento de los sistemas para diferentes casos de uso. De este entregable se extrae que el consumo energético depende de muchos factores y es difícil de caracterizar. De hecho, el consumo energético no solo depende de la configuración de la vBS y de su carga, si no que también depende del hardware donde se despliega dicha estación base. Dicho esto, y a pesar de que los valores absolutos de potencia medido cambian dependiendo de la plataforma hardware, se encuentra patrones similares en todas las mediciones. Por ejemplo, el consumo energético aumenta para esquemas de modulación (MCS) más altos (ver SORUS-RAN-A2.1-E1 para más información).

Sin embargo, en el anterior entregable no consideramos los efectos que tiene la agregación de vBS sobre el consumo energético y el rendimiento. Esta dimensión es muy importante ya explota la flexibilidad de la virtualización y permite compartir recursos y por lo tanto ahorrar costes, uno de los temas pendientes en redes virtualizadas. Es por ello, que en este entregable con nos enfocamos en sistemas de recursos compartidos e interferencia entre los procesos de la vBS.

El documento está estructurado comenzando con un contexto de las vRANs y la arquitectura O-RAN en la Sección 4. La Sección 5 aporta detalles acerca del diseño conceptual de la plataforma experimental. Se continua con el perfilado de vRAN en la Sección 6, donde se desarrollan distintos experimentos comparativos en función de parámetros de interés. Y, finalmente, se concluye el documento con las conclusiones en la Sección 7.

[16] G. Garcia-Aviles et al., “Nuberu: Reliable RAN virtualization in shared platforms,” in Proceedings of the 27th MobiCom, 2021, pp. 749–761.

[17] J. A. Ayala-Romero et al., “vrAIn: A deep learning approach tailoring computing and radio resources in virtualized RANs,” in Proceedings of the 25th MobiCom, 2019, pp. 1–16.

4 BACKGROUND

4.1. Redes de acceso radio virtualizadas (vRANs)

Es bien sabido que la PHY de una pila vBS conlleva la mayor parte de la computación[18]. A continuación, presentamos algunos antecedentes al respecto. La Figura 1 ilustra el funcionamiento de un procesador vBS PHY dúplex por división de frecuencia (FDD).

Cada 1 ms, un vBS recibe las muestras de radio asociadas a una subtrama de enlace ascendente n. Un despachador selecciona un *worker* inactivo, que inicia una cadena de tareas de procesamiento de radio en un hilo de computación independiente. Estas tareas incluyen (i) procesar los canales de datos y control transportados por la subtrama de enlace ascendente n, (ii) programar las concesiones de radio de enlace ascendente/descendente que transportará la subtrama de enlace descendente $n+M$, (iii) procesar los canales de datos y control para la subtrama de enlace descendente $n+M$, y (iv) enviar los símbolos modulados correspondientes a la subtrama de enlace descendente $n+M$ al frontend de radio. En 4G LTE, $M=4$ respecto a las restricciones 3GPP para proporcionar retroalimentación ARQ híbrida a los usuarios, pero este parámetro es configurable en 5G New Radio[19].

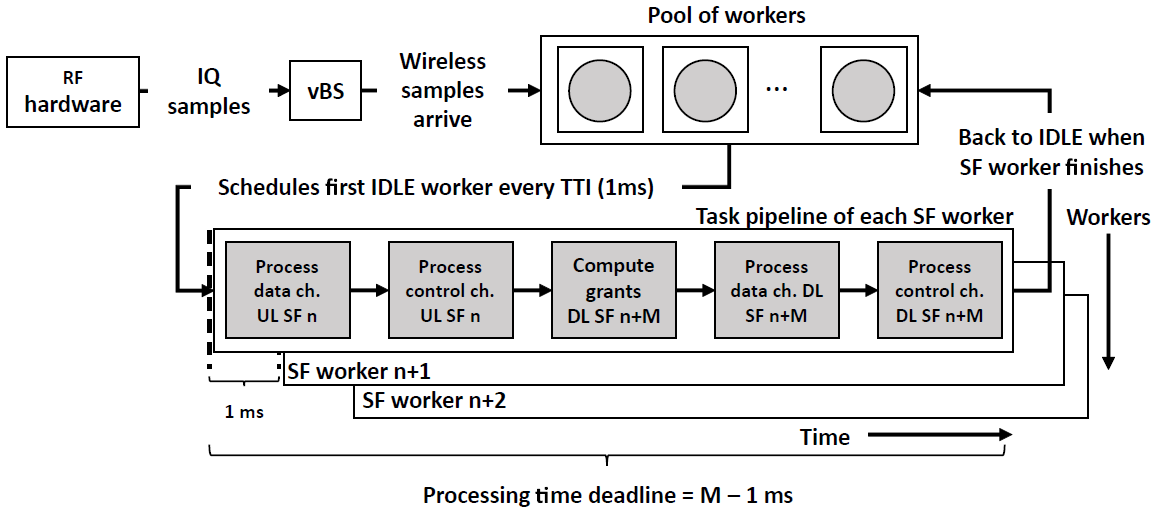

Figura 1: Cada TTI, un vBS necesita generar un nuevo subproceso para su pool para procesar las diferentes tareas de UL y DL.

[18] Y. Y. Chun, M. H. Mokhtar, A. A. A. Rahman, and A. K. Samingan, "Performance study of lte experimental testbed using openairinterface," in 2016 18th International Conference on Advanced Communication Technology (ICACT), 2016, pp. 617–622.

[19] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation," Link, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.211, 06 2022, version 17.2.0.

El procesamiento de los canales en una subtrama constituye una tarea intensiva en recursos computacionales que demanda una serie de operaciones adicionales. Estas operaciones incluyen la (de)modulación de los símbolos OFDM y las complejas operaciones de codificación de errores hacia delante (FEC), todas las cuales requieren una cantidad significativa de recursos de computación. Sin embargo, la naturaleza misma de las redes vRAN impone restricciones temporales estrictas: es necesario generar una subtrama de enlace descendente cada 1 milisegundo y procesar una subtrama de enlace ascendente cada 1 milisegundo también. Ante este escenario, surge la necesidad de garantizar que los recursos computacionales estén disponibles de manera oportuna y eficiente para llevar a cabo estas operaciones críticas.

Para optimizar el uso de los recursos disponibles y garantizar que los trabajos se ejecuten de manera eficiente, se implementa la paralelización en canal. Esto implica asignar un conjunto de $M-1$ workers para llevar a cabo los trabajos, de modo que un worker se active para ejecutar una tarea y, una vez finalizada, quede en espera de nuevos trabajos. Esta estrategia permite distribuir la carga de trabajo de manera equitativa entre los workers disponibles, garantizando así un rendimiento óptimo del sistema y evitando cuellos de botella en el procesamiento de las subtramas.

Entre las diversas tecnologías de virtualización disponibles en la actualidad, incluyendo máquinas virtuales, unikernels y contenedores, consideramos que los contenedores Docker son los más adecuados para satisfacer los exigentes requisitos de las cargas de trabajo vRAN[20]. Los contenedores Docker ofrecen una serie de ventajas clave que los hacen especialmente adecuados para este fin. En primer lugar, los contenedores Docker permiten una asignación granular de recursos en línea, lo que significa que los recursos computacionales pueden asignarse y ajustarse dinámicamente según las necesidades de la carga de trabajo en tiempo real. Esto garantiza que los recursos estén disponibles cuando y donde se necesiten, sin desperdiciar recursos innecesariamente.

Además, los contenedores Docker facilitan la orquestación de múltiples inquilinos a través de múltiples hosts. Esto significa que es posible ejecutar y gestionar eficientemente múltiples instancias de carga de trabajo vRAN en un entorno distribuido, maximizando así la utilización de los recursos disponibles y optimizando el rendimiento del sistema en su conjunto.

Otra ventaja significativa de los contenedores Docker es su capacidad para admitir la migración rápida y en vivo de contenedores. A diferencia de las máquinas virtuales, donde la migración puede ser un proceso lento y complejo, Docker permite mover contenedores de manera rápida y transparente entre hosts, lo que facilita la escalabilidad y la gestión dinámica de las cargas de trabajo vRAN. Por último, Docker ofrece una experiencia de creación, actualización y despliegue de imágenes rápida y sencilla. Esto significa que es posible crear, actualizar y desplegar nuevas

[20] A. Garcia-Saavedra and X. Costa-Perez, “O-RAN: Disrupting the virtualized ran ecosystem,” IEEE Communications Standards Magazine, 2021.

versiones de las imágenes de contenedores con facilidad y rapidez, lo que simplifica considerablemente la gestión y el mantenimiento de las cargas de trabajo vRAN a lo largo del tiempo. En resumen, los contenedores Docker ofrecen una combinación única de flexibilidad, eficiencia y facilidad de gestión que los hace ideales para soportar las exigentes cargas de trabajo vRAN. Su capacidad para proporcionar una asignación granular de recursos, orquestar múltiples inquilinos, admitir la migración rápida y en vivo de contenedores, y ofrecer una experiencia de creación, actualización y despliegue de imágenes rápida y sencilla los convierte en la opción ideal para garantizar un rendimiento óptimo y una gestión eficiente de las redes vRAN en entornos de producción.

4.2. Computación de propósito general

La Figura 2 muestra detalladamente la arquitectura de la unidad de procesamiento central (CPU) de una plataforma de computación de propósito general (GPP). En el contexto de los procesadores superescalares modernos, se aprovecha el multithreading simultáneo (SMT), también conocido como Hyper-threading en las CPU de Intel. Esta característica permite que un núcleo físico ejecute más de un hilo de ejecución a la vez. Desde la perspectiva del sistema operativo (SO), cada núcleo físico es visto como dos núcleos separados. Estos núcleos virtuales comparten el mismo procesador físico, lo que proporciona una mayor capacidad de procesamiento y eficiencia en la ejecución de tareas.La memoria caché juega un papel crucial en el rendimiento del sistema al servir como un puente entre la velocidad de acceso de la CPU y la velocidad de la memoria RAM. La memoria caché se organiza en diferentes niveles según su velocidad y tamaño[21]. La caché de nivel 1 (L1) es la más cercana y rápida del sistema, pero también la más limitada en capacidad.

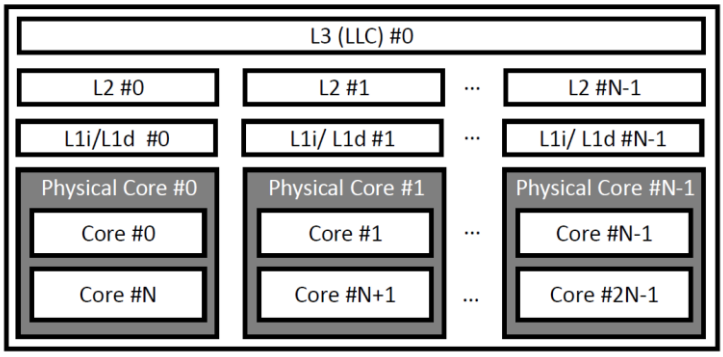

Figura 2: Arquitectura de CPU de propósito general.

[21] B. Jacob, D. Wang, and S. Ng, Memory systems: cache, DRAM, disk. Morgan Kaufmann, 2010.

Cada núcleo físico tiene su propia memoria caché L1 dedicada, lo que garantiza un acceso rápido a los datos utilizados con frecuencia. La caché de nivel 2 (L2) es de mayor tamaño que la L1 pero más lenta, y también está dedicada a cada núcleo físico. A diferencia de la L1, la L2 se utiliza principalmente para almacenar datos en lugar de instrucciones. Por último, la caché de nivel 3 (L3), también conocida como Last Level Cache (LLC), es la más lenta de la CPU y se comparte entre todos los núcleos. En una arquitectura de CPU de propósito general, cuando un núcleo ejecuta un subproceso, carga los bloques de memoria más utilizados en la caché correspondiente para garantizar un acceso rápido en futuras referencias a esos datos. Sin embargo, si un subproceso hace referencia a un bloque de memoria que no se encuentra en la caché, se produce lo que se conoce como un "fallo de caché" (cache miss), lo que activa una interrupción y obliga al núcleo a buscar los datos en una caché de memoria superior, o en última instancia, en la memoria RAM. Este proceso de búsqueda puede ralentizar temporalmente la ejecución del subproceso, pero es esencial para garantizar la coherencia de los datos y el correcto funcionamiento del sistema en su conjunto.

4.3. Arquitectura O-RAN

La O-RAN Alliance, una colaboración sin precedentes entre los principales socios industriales y operadores en el sector de las comunicaciones móviles, tiene como objetivo revolucionar las futuras tecnologías de red de acceso radioeléctrico (RAN). Su misión principal es definir un estándar técnico para la arquitectura RAN que promueva la innovación, la apertura de interfaces y la reducción de costos operativos y de despliegue mediante el aprovechamiento de la virtualización y el uso de hardware de propósito general.

La arquitectura O-RAN, representada en la Figura 3, ofrece una visión integral de este enfoque revolucionario. En su núcleo, O-RAN divide las funciones de la estación base (BS) en tres Funciones de Red (NF):

1. La Unidad de Radio (O-RU), responsable de albergar funciones de bajo nivel del físico de radio, como transformadas rápidas de Fourier (FFT) y otras operaciones relacionadas con la radiofrecuencia (RF), como amplificación y muestreo.
2. La Unidad Distribuida (O-DU), que se encarga de alojar las capas de control de enlace de radio (RLC), control de acceso al medio (MAC) y capa física de alto nivel (PHY), incluyendo operaciones de codificación y decodificación de códigos de corrección de errores (FEC).
3. La Unidad Central (O-CU) [22], subdividida en dos componentes para las funciones del plano de usuario (UP) y del plano de control (CP), que soporta los protocolos de capa superior

[22] Open RAN Alliance, “O-RAN-WG1-O-RAN Architecture Description – v04.00.00,” Tech. Spec., Mar. 2021.

como Service Data Adaptation Protocol (SDAP), Radio Resource Control (RRC) y Packet Data Convergence Protocol (PDCP).

Además de estas funciones esenciales, O-RAN también especifica la plataforma O-Cloud, diseñada para alojar Funciones de Red Virtuales (VNF) del O-gNB, proporcionando así una infraestructura flexible y escalable para la implementación y gestión de servicios RAN.

Este enfoque modular y flexible permite una mayor interoperabilidad entre componentes de diferentes proveedores y promueve la innovación al facilitar la introducción de nuevas funciones y servicios en la red. Al mismo tiempo, la adopción de estándares abiertos y la virtualización de funciones permiten una mayor eficiencia operativa y una reducción significativa de los costos de despliegue y operación de las redes móviles, sentando las bases para una industria de las comunicaciones más dinámica y competitiva en el futuro.

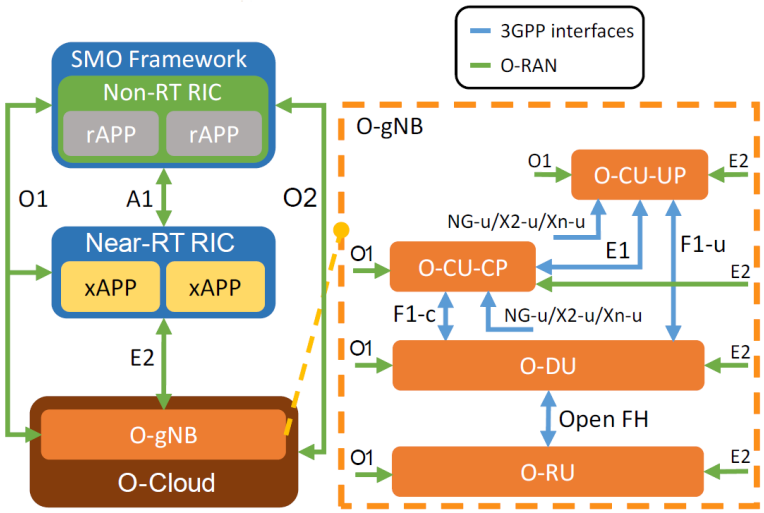

Figura 3: Arquitectura O-RAN

Para controlar y gestionar eficientemente la infraestructura O-Cloud y las funciones O-gNB, la O-RAN Alliance introduce dos controladores inteligentes de radio (RIC): el RIC en tiempo no real (non-RT RIC) y el RIC en tiempo casi real (near-RT RIC). Estos controladores desempeñan un papel fundamental en la gestión y optimización de las redes de acceso radioeléctrico (RAN) en términos de rendimiento, confiabilidad y eficiencia operativa. El marco de gestión y orquestación de servicios (SMO) sirve como alojamiento para el non-RT RIC, permitiendo la implementación de bucles de control en escalas de tiempo más amplias, como segundos o minutos. Las aplicaciones que aprovechan las capacidades de control del non-RT RIC se conocen como rApps y tienen la capacidad

de gestionar y optimizar diversos aspectos de la red, como la asignación de recursos, la configuración de parámetros y la mitigación de fallos.

Por otro lado, el near-RT RIC se encarga de gestionar bucles de control en escalas de tiempo más pequeñas, en el orden de décimas de milisegundo, a través de aplicaciones denominadas xApps. Estas xApps están diseñadas para realizar ajustes rápidos y precisos en la red, respondiendo a cambios dinámicos en las condiciones del entorno y las demandas del servicio.

La interfaz O1 desempeña un papel crucial como conexión lógica entre todos los componentes de la O-RAN y el marco SMO. Su objetivo principal es garantizar el funcionamiento y la gestión efectiva de los componentes de la O-RAN, abordando aspectos como la detección y resolución de fallos, la configuración de parámetros, la contabilidad de recursos, el monitoreo del rendimiento y la seguridad. Los componentes gestionados a través de la interfaz O1 incluyen el near-RT RIC, la O-CU y la O-DU, lo que asegura una coordinación eficiente entre ellos para optimizar el funcionamiento global de la red. Además, el near-RT RIC utiliza la interfaz A1 para recibir políticas del non-RT RIC, lo que garantiza una coordinación coherente entre ambos controladores. Por otro lado, la interfaz E2 se utiliza para recopilar información casi en tiempo real de los componentes de la O-RAN y realizar políticas de gestión de recursos de radio (RRM) de grano fino sobre ellos, lo que permite una adaptación dinámica y eficiente de los recursos de la red según las demandas del tráfico y las condiciones del entorno.

Finalmente, el SMO desempeña un papel crucial en la gestión y orquestación de la infraestructura O-Cloud a través de la interfaz O2. Esta interfaz permite una coordinación efectiva entre el SMO y los componentes de la O-Cloud, garantizando una asignación óptima de recursos y una gestión eficiente de la infraestructura para satisfacer las necesidades cambiantes de los servicios y las aplicaciones en la red. En conjunto, estas interfaces y componentes forman un marco integral para la gestión dinámica y eficiente de las redes de acceso radioeléctrico en el contexto de la O-RAN.

5 PLATAFORMA EXPERIMENTAL

La plataforma experimental que hemos desarrollado se compone de una serie de hosts que albergan los componentes necesarios para un despliegue O-RAN, así como la infraestructura de conectividad de red para los diferentes Equipos de Usuario (UEs) conectados. La Figura 4 proporciona una

representación conceptual de este banco de pruebas que hemos construido para llevar a cabo nuestras investigaciones.

En primer lugar, este banco de pruebas consta de un host dedicado que despliega el marco de gestión y orquestación de servicios (SMO) y aloja el controlador inteligente de radio en tiempo no real (non-RT RIC) (1). Además, se dispone de otro host separado que alberga la plataforma O-Cloud, donde pueden desplegarse múltiples instancias de O-eNB, y que también incluye el controlador inteligente de radio en tiempo casi real (near-RT RIC) (2). Para implementar las funciones de orquestación y gestión de la plataforma O-Cloud proporcionadas por el SMO, hemos optado por contenerizar srsRAN mediante Docker. Esta decisión nos permite aprovechar las capacidades de la API de Docker para orquestar y gestionar contenedores, facilitando así la implementación de una interfaz O2 mínima. Además, hemos empleado un agente de métricas como Telegraf para llevar a cabo la monitorización del rendimiento, lo que nos ha permitido recopilar métricas de la plataforma O-Cloud. En lugar de utilizar un orquestador comercial como Kubernetes o Docker Swarm, hemos desarrollado nuestro propio orquestador mínimo por razones de rendimiento y flexibilidad. Asimismo, hemos implementado interfaces mínimas O1 y E2 para la asignación de recursos en los vBS desplegados.

Nuestra plataforma experimental también incluye un host adicional que contiene el Evolved Packet Core (EPC) para proporcionar conectividad a los diversos UEs conectados a cada vBS (3). Dado que los vBS se ejecutan en contenedores Docker, hemos garantizado el aislamiento de las redes entre sí para evitar posibles interferencias y asegurar un funcionamiento óptimo del sistema en su conjunto.

Este diseño nos permite simular de manera efectiva un entorno de red virtualizado y evaluar el rendimiento y la eficacia de la arquitectura O-RAN en condiciones controladas.

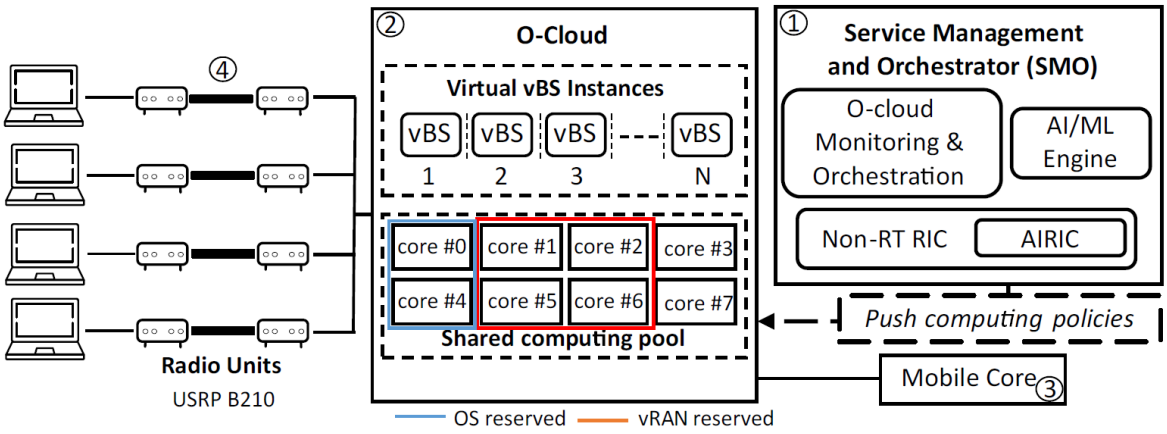

Figura 4: Diseño conceptual de la plataforma experimental

El host O-Cloud se configura en un entorno que comprende un procesador de propósito general (GPP) Intel i7-7700K equipado con 4 unidades centrales de procesamiento (CPU) físicas. En nuestro caso, hemos optado por utilizar el sistema operativo Ubuntu 20.04.5 LTS con kernel 5.13.19 para respaldar nuestras operaciones. De estas 4 CPU físicas, hemos reservado una para el funcionamiento del sistema operativo y scripts personalizados destinados a gestionar los experimentos, interactuar con la API de Docker y recopilar datos. En esencia, hemos emulado una plataforma GPP vRAN de pequeña escala con $N = 2$ CPU físicas y 4 núcleos virtuales, lo que nos permite simular de manera eficiente y realista el entorno de ejecución vRAN.

Además, como parte de nuestra infraestructura experimental, hemos integrado 4 tarjetas USRP SDR para ofrecer soporte a un máximo de 4 estaciones base virtuales (vBS), junto con los usuarios finales correspondientes que generan carga de red (4). Estas tarjetas USRP SDR se utilizan para generar flujos tanto de enlace ascendente como de enlace descendente. Para iniciar y gestionar estos flujos, empleamos la herramienta *mgen*[23], que nos permite generar flujos de datos desde o hacia los UEs y desde o hacia el EPC según sea necesario para nuestras pruebas y experimentos.

[23] HTTPS://GITHUB.COM/USNAVALRESEARCHLABORATORY/MGEN

Dado el alcance limitado de capacidad de cómputo de nuestra plataforma experimental, hemos establecido el ancho de banda de cada vBS en 10 MHz para garantizar un rendimiento óptimo y consistente en todas nuestras evaluaciones y pruebas. Esta configuración nos permite realizar experimentos bajo condiciones controladas y reproducibles, lo que facilita una evaluación precisa del rendimiento y la eficacia de la plataforma O-RAN en diferentes escenarios y condiciones operativas.

6 PERFILADO VRAN

6.1. Interferencias de recursos computacionales (noisy neighbours)

La Figura 5 representa el uso relativo de CPU del sistema en función del número de instancias vBS desplegadas. Las barras en azul muestran el uso esperado suponiendo un aislamiento perfecto de los recursos. Las calculamos escalando linealmente el uso de CPU de una única instancia vBS. Las barras rojas muestran el consumo real de CPU, que desvela una sobrecarga que crece exponencialmente inducida por la mencionada contención de recursos en plataformas de computación imperfectamente aisladas.

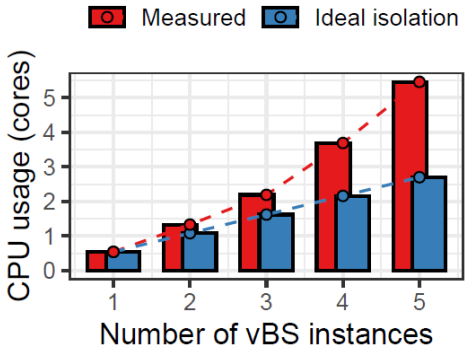

Figura 5: Uso de CPU cores en función del número de vBSs desplegadas

En el contexto de vRAN, explorar las ventajas y el impacto de la virtualización de funciones de red radioeléctrica puede suponer un reto si se tienen en cuenta las características específicas de RAN. En primer lugar, la carga de trabajo de las vBS tiene plazos estrictos, lo que los hace mucho más sensibles al problema de *noisy neighbour* que los VNF clásicos, como conmutadores o cortafuegos. Lo confirmamos en la Figura 6, que muestra el rendimiento normalizado de un VNF para diferentes asignaciones de CPU (eje x). Obsérvese que su rendimiento se desploma rápidamente en caso de déficit de recursos de computación. Esto ocurre porque se incumplen los tiempos de la capa física (PHY), lo que hace que los usuarios pierdan la sincronización con el vBS, con la consiguiente pérdida

de conectividad[24]. Esto difiere significativamente de los casos de VNFs normales, que sufren una degradación del rendimiento más suave ante la escasez de recursos de computación.

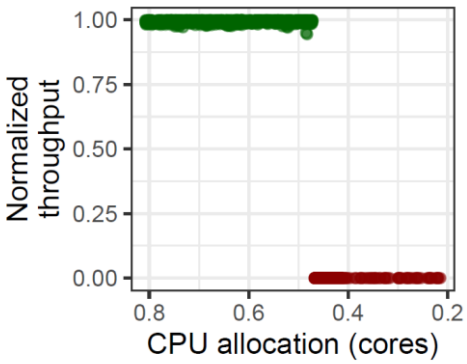

Figura 6: Throughput en función de la asignación de recursos CPU

El aumento del uso de recursos computacionales tiene su origen, entre otros motivos, en la falta de aislamiento de la memoria caché. El aumento del uso de recursos computacionales plantea un problema de aumento del consumo total de energía de la plataforma vRAN. La Figura 7 muestra la relación entre el consumo de energía normalizado sobre el consumo base del sistema (es decir, en reposo) de nuestra plataforma vRAN en función de la carga computacional total. El uso de recursos computacionales y la energía están relacionadas linealmente[25] [26]. Por lo tanto, es fundamental minimizar el uso de recursos computacionales de la plataforma vRAN con el fin de minimizar los costes energéticos operativos.

[24] G. Garcia-Aviles et al., "Nuberu: Reliable RAN virtualization in shared platforms," in Proceedings of the 27th MobiCom, 2021, pp. 749–761.

[25] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," ACM SIGARCH computer architecture news, vol. 35, no. 2, pp. 13–23, 2007.

[26] C. Lefurgy et al., "Server-level power control," in Fourth International Conference on Autonomic Computing (ICAC'07). IEEE, 2007, pp. 4–4.

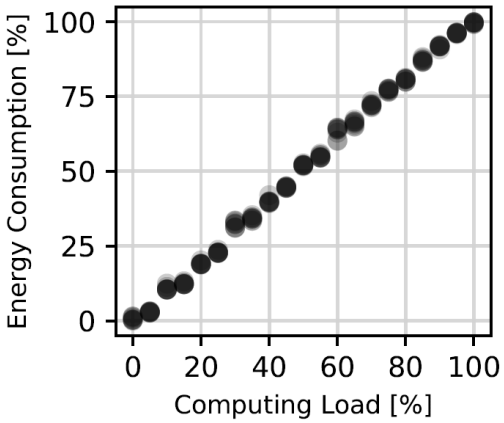

Figura 7: Consumo de energía de la plataforma vRAN en función del uso de recursos computacionales.

6.2. Aislamiento de red

Las redes virtuales conllevan una sobrecarga de recursos computacionales considerable. En el caso de los contenedores, la tecnología de virtualización empleada es una combinación de espacios de nombres de red y pares Ethernet virtuales. Con altas velocidades de transmisión de datos y paquetes de pequeño tamaño, el número de operaciones que deben procesar el host y el contenedor consume una cantidad sustancial de CPU. Se trata de un problema bien conocido y recogido en multitud de publicaciones[27] [28].

Las vBS tienen (al menos) dos interfaces de red: una interfaz con el backhaul, que conecta los vBS con el núcleo móvil (interfaces 3GPP S1/Nn [29]) y otra que conecta con otros vBS (interfaz 3GPP X2/Xn [30]). En el caso de las vRAN, la virtualización de la red no difiere de la de las VNF tradicionales. Por lo tanto, esperamos que las técnicas comunes de aislamiento de red, a través de *network namespaces*, utilizadas en NFV se comporten de manera similar.

La Figura 8 (izquierda) compara el uso medio de CPU de escenarios con 1 a 5 instancias vBS compartiendo la misma interfaz de red física para backhauling. Todas las instancias vBS son homogéneas, con frecuencias dedicadas, y saturamos su capacidad inalámbrica en las direcciones

[27] A. Tootoonchian et al., "ResQ: Enabling SLOs in Network Function Virtualization," in Proceedings of the 15th USENIX NSDI, 2018, pp. 283–297.

[28] J. Khalid et al., "Iron: Isolating network-based cpu in container environments," in Proceedings of the 15th USENIX NSDI, 2018, pp. 313–328.

[29] 3GPP, "Evolved Universal Terrestrial Radio Access Network (EUTRAN); S1 general aspects and principles," Link, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.410, 04 2022, version 17.0.0.

[30] 3GPP, "Evolved Universal Terrestrial Radio Access Network (EUTRAN); X2 Application Protocol (X2AP)," Link, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.423, 06 2022, version 17.1.0.

UL y DL. Además, en un intento de reducir otras posibles fuentes de conflicto de recursos, en este caso asignamos a cada vBS núcleos de CPU dedicados.

Probamos dos casos: (i) aislar la pila de red de los vBS individuales del host utilizando diferentes espacios de nombres de red ("virtual network"), y (ii) permitir que todos las vBS utilicen la red del host sin ningún aislamiento de espacios de nombres ("host network interface"). A partir de la figura, observamos que la sobrecarga computacional de los espacios de nombres individuales es insignificante. La razón es que la carga de red agregada generada por cada vBS es considerablemente menor que los escenarios evaluados en la literatura relacionada[31] [32] (que manejan tasas superiores al Gigabit). Por lo tanto, el aislamiento de la red no puede explicar la carga computacional mostrada en la Figura 5.

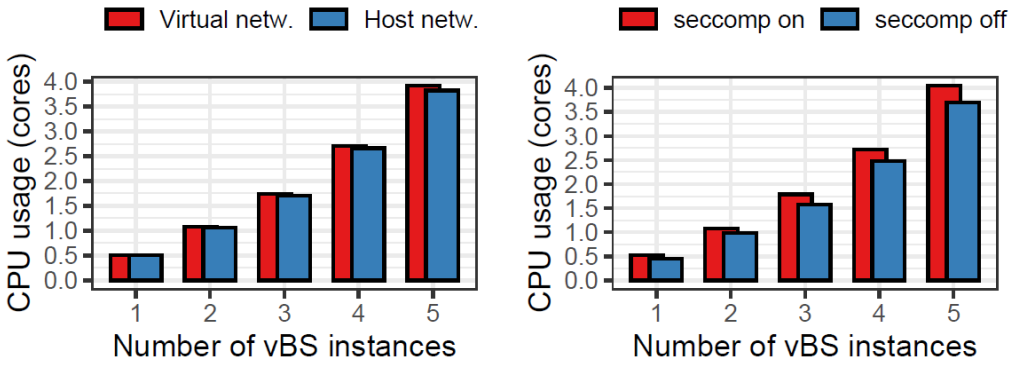

Figura 8: 95 percentil del uso de CPU agregado en una vRAN con diferente número de vBS instanciadas. Comparación virtual network vs. host network interface (izquierda) y Seccomp enabled vs. Seccomp disabled (derecha).

6.3. Filtros de computación segura

Los contenedores Docker (y otros) utilizan, por defecto en la mayoría de los GPP modernos, una función de seguridad denominada filtros de computación segura (Seccomp)[33], los filtros Seccomp pueden controlar el acceso a más de 300 llamadas al sistema (44 por defecto en Docker, lo que equilibra la protección y la compatibilidad). En el contexto de las vRAN multi-usuario, esta

[31] . Khalid et al., "Iron: Isolating network-based cpu in container environments," in Proceedings of the 15th USENIX NSDI, 2018, pp. 313–328.

[32] A. Tootoonchian et al., "ResQ: Enabling SLOs in Network Function Virtualization," in Proceedings of the 15th USENIX NSDI, 2018, pp. 283–297.

[33] T. L. man pages. (2022) Operate on secure computing state of the process.

característica adquiere una importancia capital para proteger la plataforma subyacente y mitigar posibles ataques entre arrendatarios potencialmente competidores.

Aunque la sobrecarga de los filtros seccomp está menos estudiada en la literatura, existen algunos trabajos previos que informan de un coste computacional asociado a los filtros seccomp que oscila entre < 10% (perfil seccomp por defecto en Docker) y casi 100% (con un esquema sobreprotector)[34] con aplicaciones convencionales. Para complementar ese trabajo, perfilamos el impacto de los filtros seccomp en el contexto de las vRANs.

Para ello, desplegamos los mismos escenarios utilizados en la sección anterior (utilizando interfaces de red virtuales) y medimos el uso de CPU sin filtros seccomp ("seccomp off") y con el perfil seccomp por defecto en Docker ("seccomp on"). En línea con trabajos previos**Error! Unknown switch argument.**, la Figura 8 (derecha) muestra una carga computacional extra aproximada del 1,4% en el tiempo de CPU por cada instancia vBS en el sistema, lo que suma un 7% en total con 5 instancias vBS. Se trata de una sobrecarga no despreciable, pero que no explica totalmente el gran peaje observado en la Figura 5.

6.4. Cambios de contexto

El siguiente paso natural es estudiar el impacto de los cambios de contexto (context switches) de la CPU. La contención de hilos en CPUs compartidas puede llevar a un mayor número de cambios de contexto y, en consecuencia, aumentar el consumo total de recursos de la CPU.

Para evaluar esto, repetimos los mismos escenarios que antes, y representamos en la Figura 9 el uso agregado de CPU para un número variable de instancias vBS. Al igual que antes, asignamos núcleos de CPU dedicados (CPU pinning) a instancias vBS individuales en un intento de garantizar el aislamiento de recursos. En la figura, comparamos nuestro resultado empírico con el resultado esperado con un aislamiento ideal. Aunque, como era de esperar, el impacto es considerable, sólo representa el 43% de la sobrecarga observada en la Figura 5.

[34] D. Skarlatos et al., "Draco: Architectural and operating system support for system call security," in 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2020, pp. 42–57.

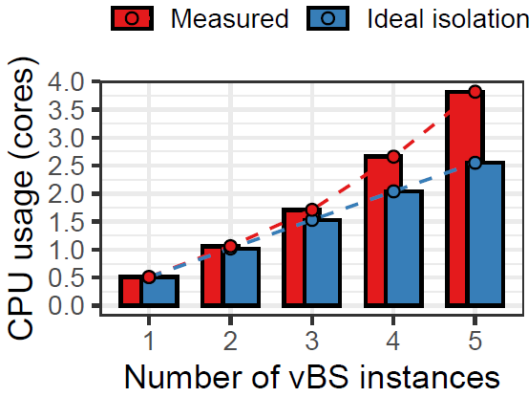

Figura 9: 95 percentil del uso de CPU agregado en función del número de vBS instanciadas y con CPU pinning.

Para obtener más información, la Figura 10 compara la proporción de cambios de contexto experimentados por un vBS individual en dos configuraciones diferentes: (i) cuando cada vBS está anclada a una CPU individual (izquierda) (ii) cuando el scheduler por defecto de la CPU es libre de asignar hilos dentro de la CPU compartida (derecha).

A partir de la Figura 10 (izquierda), observamos que el ratio de cambios de contexto permanece muy similar independientemente del número de vBSs desplegados. En este caso, toda la contención de CPU es causada por los hilos que pertenecen al vBS muestreado. Puesto que se trata de vBS homogéneas (que implementan la misma cantidad de hilos), y cada uno de ellos está anclado a una CPU dedicada, la cantidad de contención en CPU individuales es independiente del número de vBS desplegados.

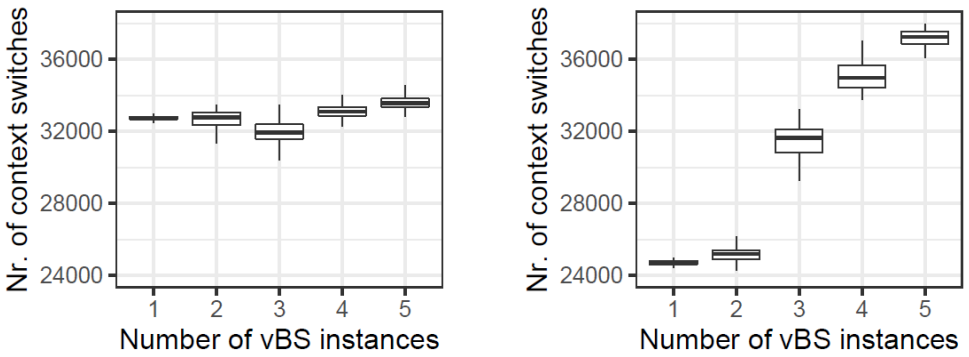

Figura 10: Cambios de contexto por milisegundo para una vBS con pinning (izquierda) y sin pinning (derecha).

Observamos un comportamiento diferente en la Figura 10 (derecha). En este caso, los hilos de todos los vBS compiten por el mismo pool de CPUs. Sorprendentemente, cuando el número de instancias vBS desplegadas en la plataforma es 1 ó 2, el ratio de cambios de contexto es menor que cuando los vBS utilizan CPUs dedicadas. La razón es que el número de instancias (1 o 2) es relativamente menor que el número de CPU en el pool (6 núcleos virtuales con $N = 3$). Por lo tanto, los hilos individuales suelen encontrar menos contención que en la configuración utilizada para la Figura 10 (izquierda) porque, allí, las CPU individuales se dedican a instancias individuales de vBS pero se comparten entre los hilos que implementan el vBS (contención intra-vBS). Por el contrario, cuando el número de instancias se aproxima al número de CPUs en el pool (4 y 5), domina la contención entre hilos vBS y el ratio de cambios de contexto supera notablemente al de CPUs dedicadas a vBS individuales. Curiosamente, cuando desplegamos 3 instancias vBS, la contención de hilos intra-vBS e inter-vBS se equilibra y el ratio de cambios de contexto es similar al caso en que los vBS se asignan a CPU dedicadas.

Con 5 instancias de vBS, medimos un aumento aproximado del 8,5% en los cambios de contexto cuando no hay pinning con respecto al uso de pinning. Además, cuando se despliega un solo vBS, se produce un descenso del 24% en el número de cambios de contexto, que no se traduce en una reducción del tiempo total de uso de la CPU. En consecuencia, el cambio de contexto no puede explicar el mencionado aumento del 43% en el consumo total de CPU observado en la Figura 5 respecto a la Figura 9, lo que nos lleva a la siguiente subsección.

6.5. Aislamiento de la memoria cache

La memoria caché es un recurso muy relevante que a menudo se pasa por alto. Aunque Docker proporciona mecanismos eficientes para particionar y aislar diferentes tipos de recursos, no proporciona funciones para particionar los recursos de memoria caché de forma eficaz. Sin embargo, las aplicaciones intensivas en caché que comparten recursos de memoria tienden a desalojar los valores de caché de los demás, lo que aumenta el número de errores de caché[35]. Tal y como se explica la sección de background, las pérdidas de caché cuestan ciclos de CPU adicionales. Si los datos no están disponibles en una caché de bajo nivel, un núcleo que ejecute un subproceso activará una señal de interrupción que detendrá su ejecución hasta que el valor correspondiente se recupere finalmente de algún recurso de memoria de nivel superior. Este coste en ciclos de CPU difiere entre tecnologías. Sin embargo, podemos deducir su orden de magnitud observando la latencia necesaria para acceder a distintos tipos de memoria. Como referencia, la **Error! Reference source not found.** muestra la latencia para acceder a diferentes niveles de caché en una arquitectura Intel Skylake.

[35] Intel CAT, “Improving real-time performance by utilizing cache allocation technology,” Intel Corporation, April, 2015

TIPO DE MEMORIA	TIEMPO DE ACCESO[36] [37] [38]	TAMAÑO
L1 CACHE	4-6 ciclos	64 KB
L2 CACHE	14 ciclos	256 KB
L3 CACHE	50-70 ciclos	2 MB
RAM	120-600 ciclos	2-4 GB

Tabla 2 : Tiempos de acceso a memoria

Para estudiar el impacto de la contención de caché en las vRANs, utilizamos la herramienta *perf* para medir la proporción de pérdidas de caché, ciclos de CPU e instrucciones requeridas por una vBS en un sistema con 1 a 5 instancias de vBS. Estas mediciones se resumen en las Figura 11, que muestran, respectivamente, las instrucciones ejecutadas por ciclo (IPC), y el número de pérdidas de caché por cada 1000 instrucciones (MPKI). Ambas métricas muestran una alta correlación.

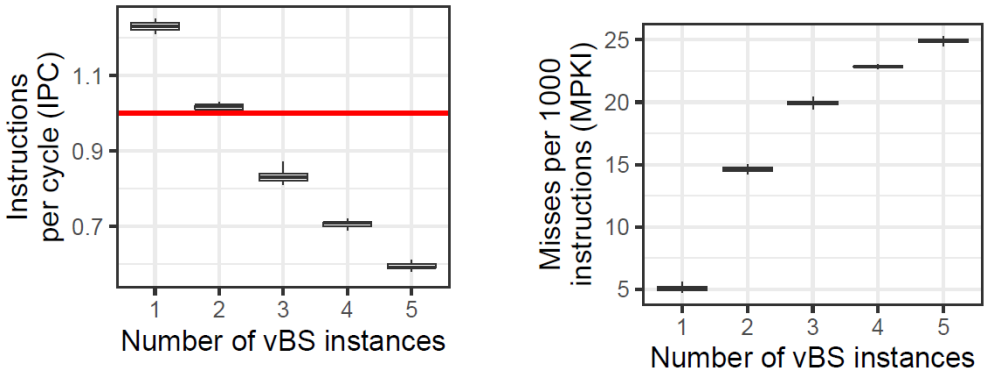

Figura 11: Instrucciones por ciclo (IPC) (izquierda) y perdidas de caché (cache misses) por cada 1000 instrucciones (MPKI) para una vBS

La Figura 11 (izquierda) evidencia que un número creciente de instancias de vBS tiene un enorme impacto en la eficiencia de cálculo. La línea roja indica un punto límite de funcionamiento en el que el sistema procesa 1 instrucción por ciclo[39]. Por un lado, cuando IPC > 1, la aplicación está limitada por instrucciones, es decir, sólo mejorando la eficiencia del código de software se puede mejorar aún más el rendimiento IPC. Por otro lado, cuando el IPC < 1, es probable que la aplicación esté limitada por un cuello de botella al acceder a recursos distintos de la CPU, como la memoria. En el

[36] B. Jacob, D. Wang, and S. Ng, Memory systems: cache, DRAM, disk. Morgan Kaufmann, 2010.

[37] J. Patterson, "Modern microprocessors: A 90 minute guide!" Cortex, vol. 15, p. A57, 2003.

[38] U. Drepper, "What every programmer should know about memory," Red Hat, Inc, vol. 11, p. 2007, 2007

[39] B. Gregg, Systems performance: enterprise and the cloud. Pearson Education, 2014.

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

caso de la Figura 11 (izquierda) esto último ocurre para un número de instancias de vBS superior a 2. Dicho cuello de botella es notable, ya que sólo permite 0,6 instrucciones por ciclo cuando se instancian 5 vBS.

Por el contrario, la Figura 11 (derecha) muestra un crecimiento notable de las pérdidas de caché por instrucción, un aumento del 500% con 5 vBSs respecto a 1. Esto, y la fuerte correlación entre las pérdidas de caché y la dinámica de IPC, nos llevan a inferir que la memoria caché es el cuello de botella en nuestro sistema vRAN y, en última instancia, la causa raíz del comportamiento anómalo de la CPU mostrado en la Figura 5.

Existen mecanismos que pueden aliviar el impacto de la contención de la caché en el consumo de la CPU. Quizás el enfoque más efectivo sea la tecnología Intel Cache Allocation Technology (CAT)[40], que nos permite particionar los recursos de memoria caché entre diferentes aplicaciones. Desafortunadamente, las tecnologías de virtualización estándar basadas en cgroups (como los contenedores Docker) no soportan dicho mecanismo de forma nativa. Por lo tanto, necesitamos encontrar estrategias alternativas que asignen recursos de CPU a las instancias vBS considerando el impacto del noisy neighbour. Exploraremos estrategias novedosas para resolver estos problemas en el entregable SORUS-RAN-A2.3-E2.

6.6. Uso de Cache L3

A continuación, estudiamos cómo afecta la asignación de las vías de caché LLC al uso de recursos de computación de una instancia vBS. En primer lugar, medimos el porcentaje de memoria caché LLC utilizado del total de memoria LLC asignada a un vBS en función de la demanda de tráfico. La Figura 12 muestra la ocupación LLC con 12 y 2 LLC líneas de cache para diferentes valores SNR y demandas de tráfico en UpLink (UL) y DownLink (DL). La serie *high* representa la ocupación LLC con un entorno SNR alto, mientras que la serie *low* representa la ocupación LLC con un entorno SNR bajo. Podemos ver que la ocupación LLC total es superior a 80% para UL y DL. Además, la ocupación de la LLC es casi ortogonal a la demanda del vBS, produciéndose un aumento de entre 2% y 6% cuando la demanda pasa del 10% a 100%.

[40] Intel CAT, “Improving real-time performance by utilizing cache allocation technology,” Intel Corporation, April, 2015, Link.

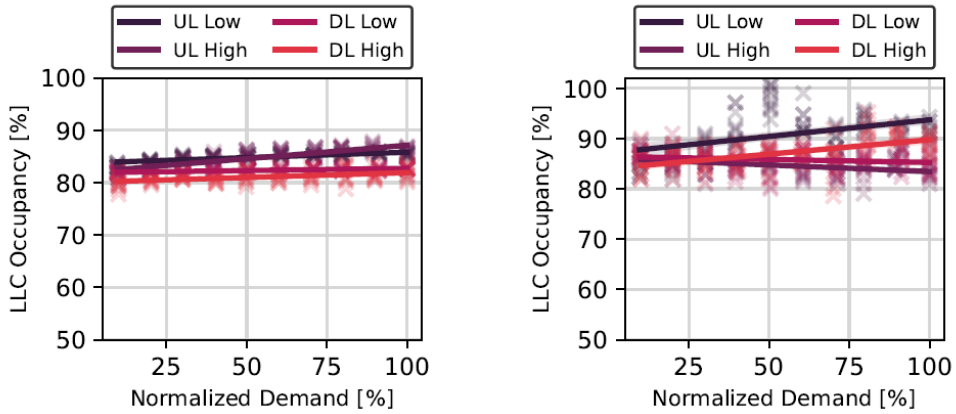

Figura 12: Uso de LLC en % en función de la demanda (eje X) para diferentes SNR (Low, High) en Uplink (UL) y Downlink (DL) considerando 12 líneas de cache (izquierda) y 2 líneas de cache (derecha)

Por último, la Figura 13 muestra el uso computacional de un vBS en función de las vías de caché L3 cuando lo desplegamos utilizando 3 núcleos de CPU. Figura 13 representa el uso de la computación para los diferentes esquemas de codificación de modulación (MCS) con una demanda de tráfico baja (es decir, el 20% de la demanda total) y alta demanda de tráfico (es decir, el 100% de la demanda total) para UL y DL. Observamos que hay diferencias significativas en la reducción del uso de recursos computacionales alcanzable. Para un tráfico elevado tanto en el enlace ascendente como en el descendente, podemos conseguir una reducción del uso de computación más significativa.

Por el contrario, una vBS que procese una demanda de tráfico baja puede obtener menores ganancias en términos de uso de recursos computacionales. Llegamos a la conclusión de que los recursos LLC tienen diferente impacto dependiendo del contexto del vBS. De esta forma, una asignación estratégica de las líneas LLC puede llevar a una reducción total del uso de recursos computacionales. En el entregable SORUS-RAN-A2.3-E2 abordamos este problema proponiendo estrategias novedosas basadas en aprendizaje máquina.

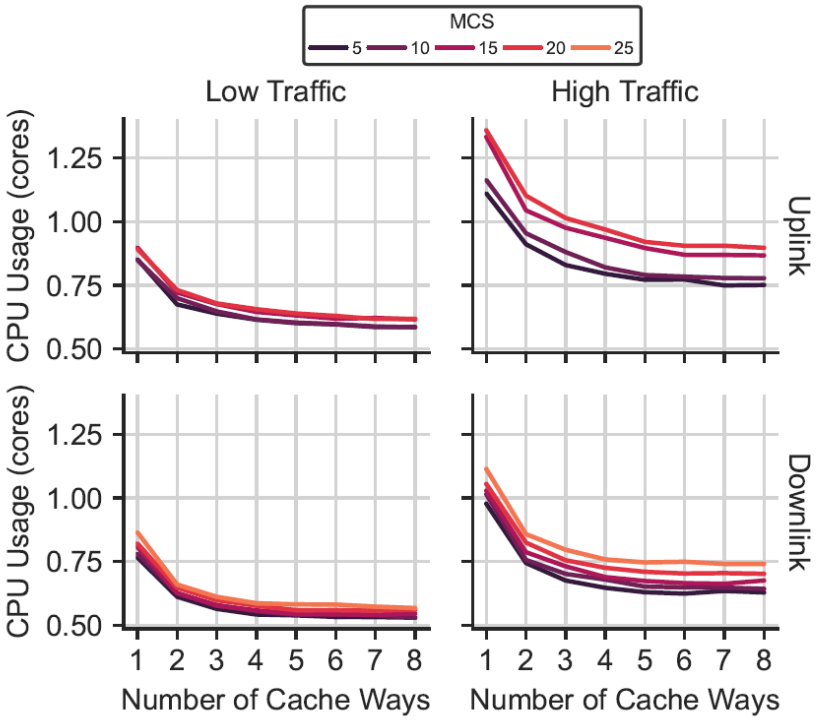

Figura 13: Uso de recursos computacionales en función de las líneas de LLC asignadas para diferentes valores de SNR y para UL y DL.

7 CONCLUSIONES

Los despliegues basados en vRAN, aunque ofrecen beneficios significativos en flexibilidad y eficiencia en el uso de recursos, no están exentos de desafíos, especialmente cuando se trata de utilizar recursos computacionales compartidos. La compartición de recursos conlleva un alto riesgo de afectar negativamente al rendimiento o de aumentar los costos de despliegue, lo que podría comprometer la viabilidad de esta tecnología en la práctica. Es crucial abordar estos desafíos para garantizar el éxito y la aceptación generalizada de la vRAN en entornos reales.

Por esta razón, en este trabajo se presenta una plataforma que busca proporcionar aislamiento de red basado en contenedores Docker. La implementación de esta plataforma se ha realizado con el objetivo de mitigar los problemas asociados con la compartición de recursos y mejorar así el rendimiento y la eficiencia de los despliegues vRAN. Se han llevado a cabo una serie de experimentos con el fin de comprender mejor las causas del aumento exponencial, en lugar de lineal como se esperaría idealmente, del uso de CPU a medida que incrementamos el número de instancias de vBS.

Los resultados de estos experimentos han sido reveladores. Se ha observado que, al considerar cinco instancias de vBS, se produce una sobrecarga aproximada del 50% en el uso de CPU. Sin embargo, también se ha identificado que la activación de la computación segura (seccomp) conlleva un

incremento del uso de CPU del 7% en comparación con el caso ideal. Además, se ha encontrado que los cambios de contexto representan el 43% de la sobrecarga total, lo que destaca la importancia de optimizar estos procesos para mejorar la eficiencia general del sistema.

Sin embargo, uno de los hallazgos más significativos ha sido la observación de que el problema principal reside en la gestión de la memoria. Cada nueva instancia de vBS provoca un aumento significativo en las pérdidas de caché por instrucción, llegando a alcanzar un aumento del 500% con cinco vBS. Este descubrimiento subraya la necesidad de abordar de manera efectiva este problema para mejorar el rendimiento general del sistema vRAN.

En respuesta a estos desafíos, se ha decidido emprender un estudio detallado y proporcionar soluciones que tengan en cuenta el "noisy neighbor" para mitigar el impacto de las pérdidas de caché en el rendimiento. Se explorarán diversas estrategias y técnicas con el objetivo de suavizar estas repercusiones y mejorar así la eficiencia y la viabilidad de los despliegues vRAN en entornos prácticos y reales. Este trabajo representa un paso importante hacia adelante en la comprensión y la optimización de la virtualización de la RAN y tiene el potencial de impulsar aún más el desarrollo y la adopción de esta tecnología en el futuro.

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810