CEA Comité Español de Automática
XLV JORNADAS AUTOMÁTICA MÁLAGA 2024
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Entorno basado en contenedores Linux para el desarrollo de aplicaciones robóticas

Ambrosio-Cestero, G.*, Matez-Bandera, J.L., Ruiz-Sarmiento, J.R., Gonzalez-Jimenez, J.

Grupo de Percepción Artificial y Robótica Inteligente (MAPIR), Dept. de Ingeniería de Sistemas y Automática, Instituto Universitario en Ingeniería Mecatrónica y Sistemas Ciberfísicos (IMECH.UMA), Universidad de Málaga, Blvr. Louis Pasteur, 35, 29071 Málaga, España.

To cite this article: Ambrosio-Cestero, G., Matez-Bandera, J.L., Ruiz-Sarmiento, J.R., Gonzalez-Jimenez, J. 2024. Linux container-based environment for the development of robotic applications. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10943

Resumen

El desarrollo y despliegue de aplicaciones robóticas en investigación involucra desafíos como la gestión eficiente de hardware heterogéneo, especialmente GPUs, o la elaboración de configuraciones software con requisitos incompatibles, por ejemplo, conflictos de librerías y versiones. A menudo, estos problemas se convierten en una limitación para los investigadores, ya que dificultan la colaboración o incluso imposibilitan el desarrollo y despliegue de sus aplicaciones. En este trabajo, se presenta una solución consistente en un entorno basado en virtualización mediante contenedores persistentes de baja latencia, que ofrece plataformas de desarrollo completos, acceso directo al hardware y gestión automática de las comunicaciones, facilitando el desarrollo de aplicaciones robóticas en entornos heterogéneos complejos. El entorno propuesto se valida mediante su implementación real en un laboratorio de robótica. Concretamente, se presenta un experimento consistente en la creación de mapas semánticos con robots móviles, una tarea compleja que ha requerido el uso de contenedores que ejecutan nodos de ROS2 intercomunicados.

Palabras clave: Robótica inteligente, Robótica conectada, Construcción de mapas, Aprendizaje automático, Multiagentes.

Linux container-based environment for the development of robotic applications

Abstract

The development and deployment of robotic applications in research involve challenges such as the efficient management of heterogeneous hardware, especially GPUs, or the creation of software configurations with incompatible requirements, for example, library and version conflicts. These challenges often become a major limitation for researchers, by hindering the collaboration or even making the development and deployment of their applications impractical. In this paper, we present a solution consisting of an environment based on virtualization through low-latency persistent containers. This environment offers complete development environments, direct access to hardware, and automatic communication management, easing the development of robotic applications in complex heterogeneous environments. The proposed environment is validated through its real implementation in a robotics laboratory. Specifically, we present an experiment involving the building of semantic maps with mobile robots, a complex task that required the use of containers running ROS2 nodes that communicate with each other.

Keywords: Intelligent Robotics, Networked Robotic Systems, Map building, Machine Learning, Multi-agent Systems.

1. Introducción

El desarrollo y despliegue de aplicaciones robóticas es fundamental para dotar a los robots con los mecanismos necesarios para operar en su entorno de trabajo. Esto incluye desde funciones básicas, como navegar de manera segura, hasta tareas más avanzadas, como la interacción con elementos del entorno. Generalmente, estas aplicaciones robóticas son desarrolladas de manera aislada en equipos con un hardware dedicado, preparados para satisfacer las necesidades específicas de dicha aplicación. Este escenario es habitual donde, a me-

*Autor para correspondencia: gambrosio@uma.es

nudo, cada investigador desarrolla sus propias aplicaciones de manera independiente. Sin embargo, este tipo de escenario inevitablemente lleva implícito dos grandes limitaciones: i) falta de flexibilidad para construir aplicaciones más complejas a partir de la combinación de aplicaciones individuales y ii) uso ineficiente del hardware que suele ser dedicado.

La primera limitación surge cuando se busca implementar un conjunto de aplicaciones robóticas para realizar tareas complejas. Esto requiere de la integración de múltiples soluciones software realizadas por terceros con bajo grado de adaptabilidad a los cambios en las especificaciones (por ejemplo, diferentes versiones de una misma librería) (Hafi et al., 2022). Esto da lugar a complicados puzzles de requisitos que no tienen solución. La segunda limitación está relacionada con el uso del hardware, en especial GPUs, por su gran relevancia en el despliegue de aplicaciones basadas en redes neuronales (Wu et al., 2023). Por lo general, este hardware es dedicado para cada equipo y se utiliza muy por debajo de su capacidad máxima, lo que conlleva un uso ineficiente del mismo. Estas dos limitaciones llevan a una reflexión: *¿tiene sentido tener equipos específicamente configurados con un hardware dedicado?*

En su lugar, parece adecuado disponer de un sistema donde se pueda realizar de manera flexible múltiples configuraciones software, lo que permitiría la coexistencia de aplicaciones robóticas heterogéneas, al tiempo que el hardware fuese compartido para así optimizar su uso. En esta dirección es posible encontrar algunas alternativas que se utilizan para salvar en parte esta problemática. Para el caso de las configuraciones específicas, es habitual recurrir a la *virtualización* para la provisión de entornos independientes que pueden ser adaptados según las necesidades concretas de cada aplicación, a partir de un interfaz virtual con el hardware. La virtualización permite cierta independencia del hardware subyacente y la posibilidad de disponer de entornos virtuales e independientes. Algunas soluciones populares son VirtualBox de Oracle, VmWare, o Hyper-V de Microsoft.

Es todavía más habitual si cabe encontrarnos con la tecnología Docker (Merkel, 2014) y Kubernetes. Esta tecnología privada, aunque de acceso público, se basa en el concepto de *contenedor*, que se define como un paquete de software autocontenido que integra todos los elementos software necesarios para llevar a cabo su funcionalidad, incluyendo al propio sistema operativo. Por otro lado Kubernetes, una plataforma de orquestación de contenedores, ofrece varias ventajas sobre Docker cuando se trata de gestionar aplicaciones en contenedores a escala.

Las aplicaciones robóticas deben (Malavolta et al., 2021; Macenski et al., 2022): 1) permitir la creación de múltiples entornos configurables, 2) disponer de acceso directo al hardware que permita operar con baja latencia, 3) posiblitar el estado persistente de los entornos y 4) gestionar y facilitar las comunicaciones entre diferentes entornos. Si bien Docker y Kubernetes ofrecen beneficios significativos en términos de la creación de entornos configurables y la gestión de aplicaciones a escala, las aplicaciones robóticas tienen requisitos específicos que pueden no ser satisfechos plenamente por estas tecnologías. Las limitaciones en el acceso al hardware en tiempo real, la compatibilidad con drivers, la complejidad de las configuraciones de red y las demandas de rendimiento son factores críticos que deben considerarse

En este trabajo se presenta un entorno basado en contenedores Linux (Graber, 2014) para el desarrollo y despliegue de aplicaciones robóticas, los cuáles emplean la tecnología de los contenedores virtuales (Soltesz et al., 2007), que satisface las necesidades mencionadas previamente. Concretamente, nuestra propuesta facilita y gestiona la creación de contenedores Linux persistentes, lo que permite la creación de configuraciones software de manera flexible. Además, el entorno otorga a los contenedores acceso directo al hardware compartido, que a su vez es controlado de manera automática por el sistema, aliviando y facilitando el trabajo del investigador. Así mismo, las comunicaciones entre los diferentes contenedores, independientemente de la máquina física en la que se encuentren alojados, están gestionadas por el propio sistema. Por último, dado que el objetivo es dar soporte a aplicaciones robóticas, el sistema está completamente preparado para trabajar con ROS2 de manera nativa. Esto permite al investigador desarrollar y desplegar aplicaciones robóticas de manera tradicional, pero con la ventaja de que cada aplicación se ejecuta en un contenedor con su configuración específica, optimizando el uso del hardware y evitando los problemas habituales de incompatibilidades software y uso ineficiente de recursos.

Finalmente, el entorno presentado en este trabajo se valida con un caso de uso real: la de creación de mapas semánticos con un robot móvil. Para ello, múltiples aplicaciones implementadas como nodos de ROS2 se ejecutan simultáneamente con unas especificaciones de software y hardware específicas, existiendo incompatibilidades entre ellas. Los resultados demuestran la efectividad de la propuesta, permitiendo a los investigadores desarrollar sus algoritmos evitando potenciales incompatibilidades del software, a la vez que se promueve la utilización eficiente del hardware.

2. Trabajos relacionados

El desarrollo de sistemas robóticos requiere de conocimientos y habilidades interdisciplinares que buscan la integración de hardware y software en sistemas complejos. Estas tareas las llevan a cabo diferentes investigadores o equipos cuya coordinación resulta crítica (Hafi et al., 2022). En este sentido, han surgido nuevas tecnologías basadas en contenedores que permiten un cierto tipo de virtualización que buscan proporcionar entornos consistentes y reproducibles para el desarrollo de software.

Algunas tecnologías como Docker (Merkel, 2014), ofrecen soluciones robustas mediante el encapsulado completo de entornos de software incluyendo sus dependencias, en contenedores portables. Este enfoque asegura que el software se ejecuta correctamente con independencia del sistema subyacente evitando los problemas habituales que se producen cuando las aplicaciones se intentan ejecutar fuera de los entornos en las que fueron creadas (Saha and Dasgupta, 2018). La utilización de contenedores Docker permite incluso la aplicación de técnicas de orquestación (Lumpp et al., 2023) para el desarrollo de aplicaciones basadas en ROS.

La incorporación de técnicas basadas en contenedores en el flujo de trabajo de desarrollo de software robótico también permite seguir los principios de DevOps (Jabbari et al., 2016),

en especial los de la integración continua, el despliegue continuo y las pruebas automatizadas (da Silva et al., 2023). En el ámbito de la robótica estos principios de integración mejoran la agilidad y fiabilidad del proceso de desarrollo, facilitando la entrega frecuente de nuevas funcionalidades a la vez que se disminuyen los riesgos relativos al despliegue (Ronanki, 2021). Además de estas ventajas, este enfoque permite un alto nivel de abstracción frente a las complejidades de los diferentes entornos de desarrollo, las dependencias con el resto del sistema y las configuraciones debidas al hardware, liberando al investigador de muchos de estos problemas y permitiéndole aprovechar mejor sus esfuerzos para desarrollar su idea principal (Mavrinac, 2023).

La utilización de la tecnología de contenedores en el desarrollo de software robótico ha sido ya probada en diferentes escenarios y plataformas demostrando su versatilidad y efectividad. En (White and Christensen, 2017) se tienen en cuenta las ventajas de la plataforma ROS (Macenski et al., 2022) en cuanto a su dinamismo y su modularidad, pero también sus problemas debidos a la distribución de funciones en nodos. No obstante, soluciones como Rorg (Wang et al., 2019) han demostrado mejoras significativas en la gestión de recursos y el mantenimiento del software mediante orquestación de microservicios. La integración de los paradigmas de computación en la nube y computación en el borde (del inglés *edge computing*) con el desarrollo de software robótico mediante contenedores ha mostrado resultados prometedores en la mejora de la escalabilidad y la eficiencia de los recursos (Pontarolli et al., 2023; Zhang et al., 2022).

En cualquier caso, las distintas iniciativas responden a la tecnología del momento y a situaciones muy concretas que aportan experiencias pero no soluciones adecuadas al entorno robótico. Todas ellas descansan sobre tecnología Docker, quizás porque ha sido la más robusta en los momentos en los que los trabajos fueron realizados. En la solución planteada, al tratarse de aplicaciones robóticas, se requiere una latencia muy baja, al tiempo que un entorno que facilite plataformas de desarrollo completas, lejos de los microservicios. Por ello se ha optado por otra tecnología: los contenedores Linux con las adaptaciones pertinentes.

3. Descripción del entorno propuesto

El entorno de contenedores propuesto descansa sobre la tecnología de virtualización que el propio sistema operativo Ubuntu proporciona bajo el sistema LXC (Project, 2024b). Este sistema se basa en fuentes abiertas, aunque recientemente, Canonical, la empresa matriz de Ubuntu, lo ha incorporado como parte de su sistema operativo, dejando vía libre a una nueva alternativa denominada Incus (Project, 2024a).

Desde una perspectiva de sistemas, el entorno se basa en un servidor Linux en el que cada investigador es libre de crear cuantos contenedores necesite con la configuración que crea conveniente sin más limitación que el sentido común bajo un ambiente colaborativo y conocedor de los límites de los recursos disponibles. Puesto que la mayoría de los contenedores acaban implementando nodos ROS2 que necesitan comunicarse con otros nodos de otros contenedores, cada usuario recibe un espacio de direcciones IP que se corresponde con una subred de clase A, donde el tercer byte coincide con el identificador del usuario en el sistema operativo, de modo que a cada usuario se le asigna un espacio de direcciones en la red `10.1.<id>.0/24`.

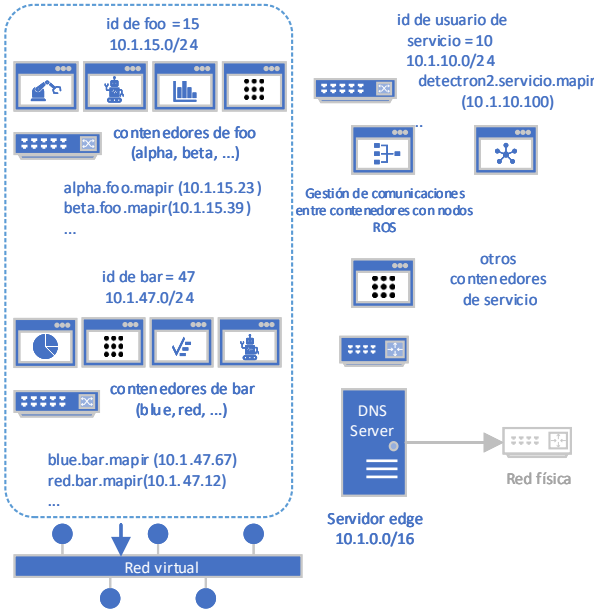

Figura 1: Ejemplo de un diseño de red virtual basada en contenedores.

La Fig. 1 muestra un ejemplo de este direccionamiento que constituye una red virtual gestionada por el servidor que, según este esquema actual, puede gestionar un máximo de 254 usuarios que, a su vez, pueden direccionar 254 contenedores cada uno. En este diseño de red virtual, cada usuario funciona como un único dominio de difusión (Capa 2 en el modelo OSI), de forma que los nodos ROS2 de cada usuario se anuncian y son visibles por el resto de los nodos que se reparten entre los contenedores de ese usuario. Esta función la proporciona el propio servidor que, para ello, actúa de conmutador virtual.

Una característica común de este tipo de tecnología basada en contenedores es que garantiza la seguridad y la independencia de los usuarios evitando la interacción entre recursos. Sin embargo, el entorno propuesto busca justo lo contrario, ya que se pretende que el software preste sus servicios como nodos ROS2, y que a su vez, estos se comuniquen entre sí. Para lograrlo, se ha modificado el *router* virtual del sistema de contenedores para permitir la comunicación entre diferentes subredes o dominios de difusión. De esta forma los contenedores de cada usuario pueden comunicarse con los contenedores de otros usuarios y, por ende, los nodos ROS2 dentro de esos contenedores.

ROS2 se distingue por introducir el Servicio de Distribución de Datos, (DDS, de sus siglas en inglés, *Data Distribution Service*) (Zhang et al., 2022; Macenski et al., 2022; Wendt and Schüppstuhl, 2022) para la comunicación entre nodos con latencia ultra baja, alta fiabilidad y gran escalabilidad. En este sistema, los nodos se anuncian en una red local mediante la transmisión de mensajes en el dominio local. Sin embargo, este tipo de tráfico de difusión no llega a otras redes locales. Para habilitar la comunicación entre nodos de ROS2 pertenecientes a diferentes redes de usuarios, el sistema dispone de un contenedor que actúa como servidor de descubrimiento

que actúa como intermediario y que facilita la comunicación al recopilar y distribuir la información de todos los participantes conectados a diferentes redes mediante una base de datos permanentemente actualizada.

Además, la solución propuesta también proporciona un sistema de nombres de dominio para facilitar la dirección de los diversos contenedores sin necesidad de conocer sus direcciones IP. Cada contenedor tiene un nombre en la forma `<contenedor>.<usuario>.<grupo>`. El servidor proporciona el servicio DNS. La Fig. 1 muestra un ejemplo con nombres de contenedores y sus direcciones IP asociadas.

En cuanto a los aspectos de acceso de los usuarios, la solución presentada utiliza el protocolo `ssh` que proporciona un entorno de comunicación seguro de acceso tanto local como remoto. Además, los usuarios en el servidor también disponen de un interfaz gráfica `xfce` a través del protocolo `RDP` que garantiza el acceso gráfico desde sistemas Linux, a través de, por ejemplo, *Remmina*, o desde sistemas Windows mediante *Remote Desktop*. Mediante este interfaz o mediante el reenvío del protocolo `X11` a través de `ssh`, garantizamos que la salida gráfica de las aplicaciones del servidor se muestren en las estaciones de trabajo desde las que se accede al sistema. Esto es de gran utilidad para las aplicaciones como *RViz* o *Rqt* muy populares en el entorno ROS2. La Fig. 2 muestra un ejemplo de acceso al entorno gráfico, que también muestra las GPU disponibles en el sistema.

Todo lo citado anteriormente se complementa con una serie de servicios que ayudan a la creación y mantenimiento de los contenedores. Entre las funciones más destacadas tenemos: las imágenes genéricas de sistemas operativos; las especializadas que incluyen drivers, librerías, y entornos de desarrollo ROS; los perfiles para la creación de imágenes; el servicio de replicación de contenedores; el servicio de creación de estados persistentes; y el servicio de copias, entre otras.

4. Implementación del entorno propuesto

Para soportar este entorno, se ha destinado un servidor físico que se ha dimensionado adecuadamente para dar servicio a un gran número de investigadores. Actualmente cuenta con un procesador i7 de nueva generación, 128GB de RAM, 4TB de almacenamiento NVME, 8TB de almacenamiento SATA para copias de seguridad, una GPU NVIDIA GeForce RTX 3060 con 8GB de memoria de vídeo y una NVIDIA Titan X con 12 GB, además de un interfaz ethernet de 2,5 Gb que permitirá su futura integración en un cluster con otros servidores. Este servidor ejecuta actualmente el sistema operativo Ubuntu Server LTS 22.04 y los drivers actualizados que gestionan las GPUs. No es necesario ningún otro software adicional ya que serán los usuarios los que, según sus necesidades, instalen sus propios paquetes de software y librerías en sus contenedores que dispondrán de sus propios sistemas operativos, compartiendo el acceso al kernel del host. Los contenedores que, como máquinas con sus propios sistemas operativos, acceden al hardware real del servidor, deben contar con los drivers adecuados.

Este servidor se encuentra conectado a una red local que comparte con el resto de equipos del laboratorio y a Internet a través del enrutamiento que porporciona la propia Universidad de Málaga. Esta conectividad permite tanto el trabajo en local como en remoto, permitiendo incluso que los nodos ROS2 puedan ser utilizados remotamente desde Internet.

Por otro lado, como en cualquier otro laboratorio, los investigadores cuentan con estaciones de trabajo o utilizan sus propios equipos (lo que se conoce como BYOD, del inglés *Bring Your Own Device*) con los que pueden acceder al servidor y construir y lanzar cuantos contenedores necesiten, sabiendo que si utilizan las imágenes preconfiguradas, contarán con entorno gráfico, entorno de desarrollo y plataforma ROS2. Además el servidor dispone de una cuenta genérica que agrupa aquellos contenedores que prestan servicios generales al resto de los usuarios como son plataformas de monitorización, obtención de métricas, gestores Docker, redes neuronales como las presentes en Detectron2 (Wu et al., 2019), etc.

En la Fig. 2 se puede observar el interfaz gráfico de un contenedor en el que se muestran una serie de ventanas con contenidos resultantes de la ejecución de una aplicación de *mapas semánticos* que se encuentra distribuida en diferentes contenedores pertenecientes a sus correspondientes investigadores (ver Sec.5). Por un lado se muestra la aplicación *RViz* de ROS2 conectada a un contenedor de otro usuario que contiene un nodo ROS2 ejecutando un *Detectron2*. Este *Detectron2* utiliza una de las GPUs, que solo alcanza el 11 % de su capacidad de ejecución. En medio se muestra una ventana con un monitor de las dos GPUs del servidor. En este caso se observa que la segunda GPU, la Titan X, no está recibiendo carga de procesamiento alguna. En la parte baja de la pantalla se observa la aplicación *rqt*, también del paquete ROS2, visualizando el mapa de nodos del sistema y la salida de las cámaras RGB y D (profundidad). Esas imágenes se están reproduciendo a partir de un dataset que está siendo leído por otro nodo situado en otro contenedor de otro investigador. Finalmente las ventanas de texto muestran la salida de otro nodo en otro contenedor de otro usuario que está aplicando el algoritmo principal que genera una nube de puntos que acaba siendo visualizada por la aplicación *RViz*.

5. Caso de uso: Construcción de mapas semánticos

Los mapas semánticos son una representación interna que poseen los robots, y que incluye información de alto nivel tanto del entorno en el que operan, como de los elementos que hay en dicho espacio, por ejemplo, la categoría de cada objeto o su funcionalidad (Ruiz-Sarmiento et al., 2017; Fernández-Chaves et al., 2021). Esta información confiere a los robots de cierta comprensión sobre el entorno, permitiendo así que puedan entender y llevar a cabo tareas más avanzadas, del tipo "*trae el limpia cristales del armario del lavadero*". Generalmente, la construcción de mapas semánticos es un proceso incremental, es decir, conforme el robot inspecciona el entorno adquiere datos a través de sus sensores, típicamente cámaras RGB-D, y los procesa para extraer el conocimiento que se almacena y fusiona a lo largo del tiempo en el mapa semántico.

De forma simplificada, este proceso requiere de la ejecución de las siguientes tareas: i) localización del robot, ii) pre-procesamiento de los datos adquiridos, iii) detección de objetos en las imágenes y iv) fusión y mantenimiento del conocimiento en el mapa semántico. Cada una de estas tareas requiere de unos recursos determinados, tanto software como hardware, que a veces son incompatibles entre sí. Por ejemplo,

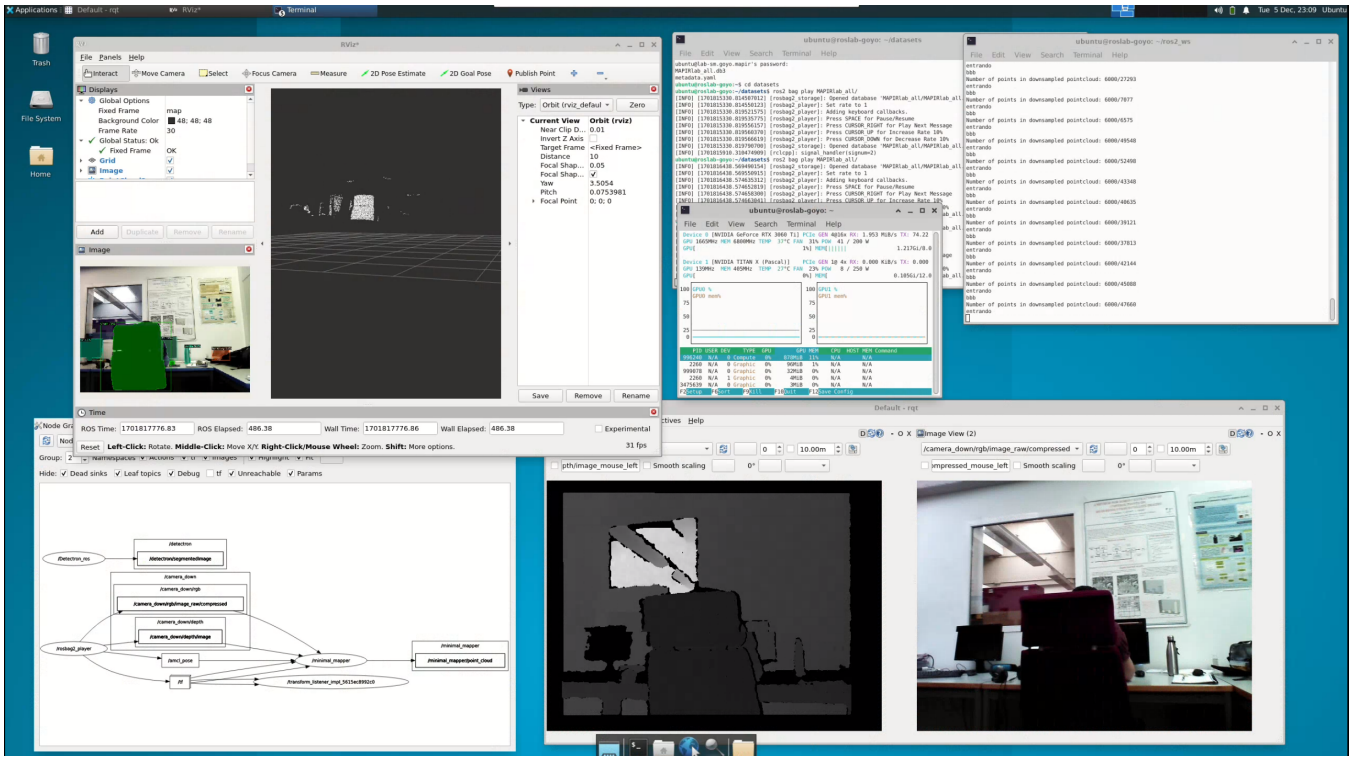

Figura 2: Interfaz gráfico de un contenedor que ejecuta una aplicación de mapas semánticos distribuida en varios contenedores.

pueden requerir versiones diferentes de una misma librería, o el propio robot puede no disponer de los recursos necesarios, como GPUs, siendo necesario realizar el procesamiento fuera de este (por ejemplo, en un servidor). En este sentido, es esencial disponer de herramientas que permitan la ejecución de aplicaciones con requisitos heterogéneos, así como la propia gestión de los recursos incluso en sistemas distribuidos.

En este caso de uso, se plantea utilizar el entorno propuesto como solución para la construcción de mapas semánticos. Concretamente, se requiere una configuración de varios contenedores con especificaciones diferentes ejecutando aplicaciones implementadas como nodos de ROS2 intercomunicadas:

- **Contenedor 1. Adquisición y procesamiento de datos.** Se encargará de la adquisición de datos y procesamiento de datos. En concreto, se encargará de obtener la localización del robot a partir de los datos obtenidos por los sensores, así como de extraer la información geométrica del entorno, por ejemplo, nube de puntos. Esta aplicación solo requiere de CPU y puede ser ejecutada localmente en el robot.

- **Contenedor 2. Detección de objetos.** Procesará la imagen RGB que recibirá del contenedor 1 mediante el uso de la red neuronal de detección de objetos presente en *Detectron2* (Wu et al., 2019). Esta red ofrece como salida una máscara indicando que píxeles pertenecen a cada objeto, así como la categoría de cada objeto objeto y la confianza que tiene sobre la detección. Esta información será enviada de vuelta al contenedor 1 para su posterior procesamiento. Esta tarea demanda el uso de GPUs, y por tanto debe ser ejecutado en el servidor *Edge*.

- **Contenedor 3. Gestión y visualización del mapa semántico.** Este contenedor se encargará de recibir del contenedor 1 cada nueva observación, que estará compuesta tanto por la localización del robot, como de la información geométrica ya anotada con el conocimiento semántico sobre los objetos. En concreto, este contenedor ejecutará *Bonxai* (Faconti, 2024), una aplicación que permite la gestión de representaciones 3D voxelizadas del entorno, donde se almacenará el conocimiento adquirido y se irá refinando mediante la fusión de nue-

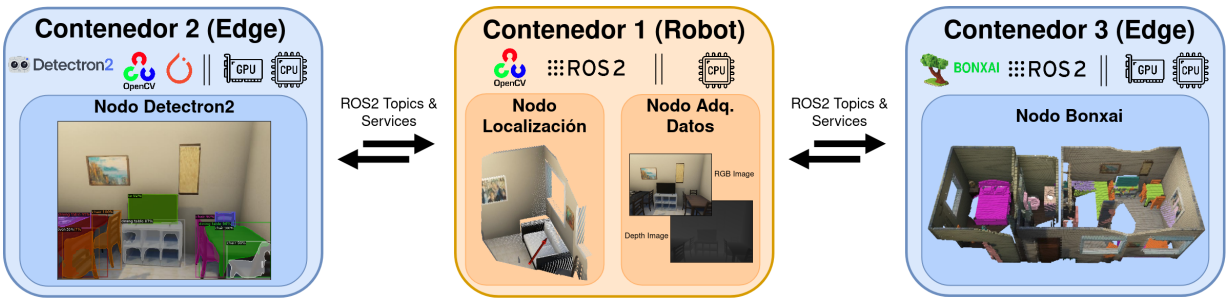

Figura 3: Esquema de los contenedores utilizados, incluyendo los requisitos software y hardware particulares, así como las comunicaciones requeridas entre ellos. Los contenedores alojados en el borde se muestran con un color de fondo azul y en naranja aquellos que se ejecutan localmente en el robot. Además, se ilustran los diferentes nodos ROS2 necesarios que se ejecutan simultáneamente para la creación del mapa semántico.

vas observaciones. Esta tarea requiere un alto uso de CPU, así como disponibilidad de GPU.

La Fig. 3 ilustra el escenario implementado para el presente caso de uso, donde en cada contenedor ha sido posible instalar el software con las versiones específicas necesarias, evitando problemas por incompatibilidades, por ejemplo, versiones de CUDA diferentes. Además, este entorno se encarga de gestionar las comunicaciones entre las máquinas físicas existentes en la red, así como los contenedores alojados en ellas. De esta manera, es posible continuar haciendo uso de las comunicaciones de ROS2 de manera natural, aún cuando los diferentes nodos de ROS2 se están ejecutando en contenedores que están alojados en máquinas físicas diferentes.

Cabe señalar que el entorno también se ha empleado con éxito en el contexto de una técnica de mapeo semántico alternativa (Moncada-Ramirez et al., 2024) y de un sistema multimodal en interacción humano-robot (Cañete et al., 2024).

6. Conclusiones

En este trabajo se propone un entorno basado en contenedores para abordar los problemas inherentes al desarrollo y despliegue de aplicaciones robóticas en entornos colaborativos y especializados. El entorno propuesto ofrece un sistema de creación y gestión automática de contenedores Linux persistentes, caracterizados por su baja latencia debido al acceso directo al hardware. Además, la propuesta gestiona las comunicaciones entre contenedores, facilitando el desarrollo de aplicaciones robóticas. De este modo, las aplicaciones con incompatibilidades pueden ser integradas en contenedores diferentes y coexistir o ejecutarse simultáneamente en una misma máquina o en la red. Finalmente, este entorno ha sido validado mediante su implementación en un laboratorio de robótica, y su posterior explotación para llevar a cabo una tarea compleja como es el mapeo semántico con un robot móvil. Su uso permitió la ejecución satisfactoria de esta tarea, la cual originalmente presentaba dificultades causadas por incompatibilidades de software y una gestión ineficiente de las GPUs.

Agradecimientos

Este trabajo ha sido desarrollado en el contexto de los proyectos ARPEGGIO (PID2020-117057GB-I00) y Voxeland (JA.B1-09), financiados por el Ministerio de Ciencia e Innovación y la Universidad de Málaga, respectivamente.

Referencias

Cañete, A., Quemada, E., Ruiz-Sarmiento, J.-R., Moreno, F.-A., Gonzalez-Jimenez, J., 2024. Multimodal system for the orientation of mobile robots towards its interlocutor. XLV Jornadas de Automática.

da Silva, A. S., Kreutz, A., Weiss, G., Rothe, J., Ihrke, C., 2023. Devops in robotics: Challenges and practices. In: Software Architecture. ECSA 2022 Tracks and Workshops. Springer International Publishing, pp. 284–299. DOI: https://doi.org/10.1007/978-3-031-36889-9_20

Faconti, D., 2024. Bonxai. https://github.com/facontidavide/Bonxai, accessed: 2024-06-03.

Fernández-Chaves, D., Ruiz-Sarmiento, J.-R., Petkov, N., Gonzalez-Jimenez, J., 2021. Vimantic, a distributed robotic architecture for semantic mapping in indoor environments. Knowledge-Based Systems 232, 107440. DOI: https://doi.org/10.1016/j.knosys.2021.107440

Graber, S., 2014. Lxc 1.0: Security features. https://www.stgraber.org/2014/01/01/lxc-1-0-security-features/, accessed: 2024-06-02.

Hafi, L. E., Ricardez, G. A. G., von Drigalski, F., Inoue, Y., Yamamoto, M., Yamamoto, T., 2022. Software development environment for collaborative research workflow in robotic system integration. Advanced Robotics 36 (11), 533–547. DOI: https://doi.org/10.1080/01691864.2022.2068353

Jabbari, R., bin Ali, N., Petersen, K., Tanveer, B., 2016. What is devops? a systematic mapping study on definitions and practices. In: Proceedings of the Scientific Workshop Proceedings of XP2016. New York, NY, USA. DOI: https://doi.org/10.1145/2962695.2962707

Lumpp, F., Panato, M., Bombieri, N., Fummi, F., apr 2023. A design flow based on docker and kubernetes for ros-based robotic software applications. ACM Trans. Embed. Comput. Syst.Just Accepted. DOI: https://doi.org/10.1145/3594539

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W., 2022. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics 7 (66). DOI: https://doi.org/10.1126/scirobotics.abm6074

Malavolta, I., Lewis, G. A., Schmerl, B., Lago, P., Garlan, D., 2021. Mining guidelines for architecting robotics software. Journal of Systems and Software 178, 110969. DOI: https://doi.org/10.1016/j.jss.2021.110969

Mavrinac, A., 2023. Devops for robotics. https://www.prosiglieres.com/posts/devops-for-robots/, [Accessed 2024-03-27].

Merkel, D., 2014. Docker: lightweight linux containers for consistent development and deployment. Linux journal 2014 (239), 2. DOI: https://dl.acm.org/doi/10.5555/2600239.2600241

Moncada-Ramirez, J., Ruiz-Sarmiento, J.-R., Matez, J.-L., Gonzalez-Jimenez, J., 2024. Large models for semantic mapping in mobile robotics. XLV Jornadas de Automática.

Pontarolli, R. P., Bigheti, J. A., de Sá, L. B. R., Godoy, E. P., 2023. Microservice-oriented architecture for industry 4.0. Eng 4 (2), 1179–1197. DOI: https://doi.org/10.3390/eng4020069

Project, L., 2024a. Incus - next-generation system container and virtual machine manager. https://linuxcontainers.org/incus/, accessed: 2024-06-02.

Project, L., 2024b. Lxc - linux containers. https://github.com/lxc/lxc, accessed: 2024-06-02.

Ronanki, K. C., 2021. Robotic software development using devops. URL: https://urn.kb.se/resolve?urn=urn:nbn:se:bth-22024

Ruiz-Sarmiento, J.-R., Galindo, C., Gonzalez-Jimenez, J., 2017. Building multiversal semantic maps for mobile robot operation. Knowledge-Based Systems 119, 257–272. DOI: https://doi.org/10.1016/j.knosys.2016.12.016

Saha, O., Dasgupta, P., 2018. A comprehensive survey of recent trends in cloud robotics architectures and applications. Robotics 7 (3). DOI: https://doi.org/10.3390/robotics7030047

Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., Peterson, L., 2007. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. ACM SIGOPS Operating Systems Review 41 (3), 275–287. DOI: https://doi.org/10.1145/1272996.1273025

Wang, S., Liu, X., Zhao, J., Christensen, H. I., 2019. Rorg: Service robot software management with linux containers. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 584–590. DOI: https://doi.org/10.1109/ICRA.2019.8793764

Wendt, A., Schüppstuhl, T., 2022. Proxying ros communications — enabling containerized ros deployments in distributed multi-host environments. In: IEEE/SICE International Symposium on System Integration. pp. 265–270.

White, R., Christensen, H., 2017. ROS and Docker. Springer International Publishing, Cham, Ch. 9, pp. 285–307. DOI: https://doi.org/10.1007/978-3-319-54927-9_9

Wu, B., Zhang, Z., Bai, Z., Liu, X., Jin, X., 2023. Transparent GPU sharing in container clouds for deep learning workloads. In: USENIX Symposium on Networked Systems Design and Implementation. pp. 69–85.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., Girshick, R., 2019. Detectron2. https://github.com/facebookresearch/detectron2.

Zhang, J., Keramat, F., Yu, X., Hernández, D. M., Queralta, J. P., Westerlund, T., 2022. Distributed robotic systems in the edge-cloud continuum with ros 2: a review on novel architectures and technology readiness. In: 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC). pp. 1–8. DOI: https://doi.org/10.1109/FMEC57183.2022.10062523