ROS-2-ARGoS Bridge: Scalable Simulations of Swarms of 1000 and More Robots

Sindiso Mkhatshwa[1,2], Tianfu Zhang[2], Paolo Leopardi[1,2], Heiko Hamann[1,2], and Andreagiovanni Reina[1,2,3]

[1] Centre for the Advanced Study of Collective Behaviour, Universität Konstanz, Konstanz, Germany
{sindiso.mkhatshwa, tianfu.zhang, paolo.leopardi, heiko.hamann, andreagiovanni.reina}@uni-konstanz.de
[2] Department of Computer and Information Science, Universität Konstanz, Konstanz, Germany
[3] Department of Collective Behaviour, Max Planck Institute of Animal Behavior, Konstanz, Germany

Abstract. The Robot Operating System (ROS) is a widely adopted collection of software libraries and tools for designing and implementing robot control software. Its rapid growth has been driven by an active community that maintains an extensive ecosystem of reusable components, increasingly positioning ROS as a unifying framework bridging academic research and industrial applications. ROS 2 importantly enables decentralized, peer-to-peer communication, making it well suited for scalable and reliable multi-robot systems. Simulations play a critical role in the design and implementation of robotic systems prior to their real-world deployment. For large-scale robotic systems, simulation scalability, that is, the efficient simulation of many robots, is essential. However, existing ROS-based simulators are unable to scale to large numbers of robots, while highly scalable simulators lack integration with the ROS ecosystem. We introduce the ROS-2-ARGoS Bridge, a framework for simulating large-scale robotic systems running software based on ROS 2. We showcase its scalability through experiments with up to 1280 simulated robots that locally interact and coordinate their actions. As industries increasingly seek to scale up their robotic infrastructures, our open-source framework offers a timely and practical solution for simulating and designing large-scale multi-robot and swarm systems.

1 Introduction

Scalability is both a feature and a challenge in swarm robotics [10]. The decentralized coordination of robot swarms avoids bottlenecks and allows for maximally scalable system design. Reported swarm experiments involve up to $N = 10^3$ robots [30,9]. Scalability is still a challenge when swarm densities (i.e., number of robots per area) are critically high, shared resources (e.g., space or communication bandwidth) deplete and the system performance is decreased due to congestion or deadlocks [14,11].

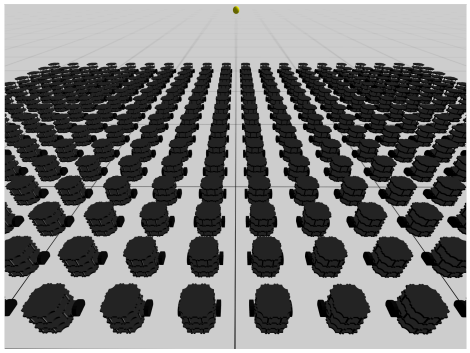

Fig. 1. A swarm of 320 simulated TurtleBot 3 robots flocking towards a light source.

Swarm robotics also has demanding requirements for the applied tooling. Commonly, swarm algorithms are iteratively developed, prototyped, and tested in robot simulators because for humans it is hard to anticipate the effects of many robot-robot interactions. This requires a 'meta-scalability,' that is, simulators of robot swarms also need to scale. Achieving system size scalability with state-of-the-art robot simulators is challenging [6]. Depending on simulation detail (e.g., robot model, physics engine), common limitations can be as low as swarm sizes of $N = 5$ or $N = 50$ robots [27]. Detailed simulations of many sensors and pairwise robot interactions increase computational costs. In addition, common robot simulators do not always allow for easy parallelization on multi-core machines.

The demand for scalable tools is growing as industry increasingly adopts concepts from swarm robotics. Although swarm robotics research emerged over 20 years ago [3], large-scale robotic systems have only seen their real-world application in recent years. For example, they are employed in warehouse automation [12] and last-mile delivery [32]. Driven by economies of scale, we anticipate further efforts to scale up future robotic systems. However, the limitations of state-of-the-art robot simulators hinder development and add overhead by necessitating custom tool implementations for each use case.

Key tools besides simulators are software frameworks that provide a modular approach, hardware abstraction, and facilitate code sharing and reuse. For robotics, ROS 2 (Robot Operating System 2) is widely adopted [16] due to its distributed, message-based software architecture. Its modular design enables seamless substitution of simulated robots with real robot hardware without altering the underlying control logic, thus ensuring portability. Moreover, ROS 2 is also well-established in industry [18], bridging the gap between research and practical applications. Hence, there is a need for a scalable robot simulator that integrates ROS 2.

We address this technological gap by integrating ROS 2 with ARGoS 3 [26], one of the most scalable robot simulators [27]. Our ROS-2-ARGoS Bridge enables efficient large-scale simulations of robotics systems (see Fig. 1). Our software allows running the robot control software (i.e., robot algorithm) using ROS 2

libraries and executing physics-based simulations in ARGoS. ARGoS simulates sensor data acquisition and updates robots' physical states (e.g., robot position) as a consequence of robots' actuation and interactions with each other and the (potentially dynamic) environment. ROS2-based control software updates robots' internal states (e.g., algorithm variables) and computes the next actions (actuation activation) based on sensor data and internal state. Our ROS-2-ARGoS Bridge enables an efficient and scalable interaction between the two components. This integration supports broader adoption of scalable simulation tools within multi-robot research and may even provide industry with a coherent substitute to ad-hoc software solutions for modern large-scale multi-robot systems.

In Sec. 2, we give an overview of existing simulation platforms. In Sec. 3, we describe the bridge implementation, its components, and its main features. Sec. 4 details the available robotic platforms in ARGoS, and the integration of sensors and actuators with the ROS-2-ARGoS Bridge. In Sec. 5, we test the scalability of the proposed system with a series of experiments with up to 1280 simulated robots. We show that our ROS-2-ARGoS Bridge allows efficient simulation of large-scale robotics, making it a promising technology for future multi-robot and swarm robotics research, as described in the concluding Sec. 6.

2 Related Work

The field of multi-robot simulator development has witnessed significant advancements in recent years, driven by the need for scalable, efficient, and portable solutions to support complex robotic systems. As the scale of robotic applications grows, the ability to simulate thousands of robots while maintaining high performance (i.e., quick simulations) and ensuring realism (useful for seamless transition from simulation to real-world deployment) has become a critical challenge. In this section, we review the state of the art in multi-robot simulation, focusing on two key aspects: scalability and development/deployment challenges.

2.1 Scalability

In multi-robot simulation, the choice of the physics engine has a decisive impact on the simulation scale and performance. Currently, mainstream simulation software, such as Gazebo [13], Webots [19], and CoppeliaSim (previously known as V-REP) [29], relies on mature 3D physics engines [33] (e.g., ODE, Bullet), which are powerful but may face performance bottlenecks when dealing with thousands of robots. PyBullet is a simulator based on the Bullet physics engine and supports GPU acceleration, which can efficiently deal with large-scale physics computation but may still be limited by computational resources for the simulation of thousands of robots [7]. Stage is a lightweight simulator limited to 2.5 dimensions (thus supporting only a restricted number and types of sensors), although it offers high simulation speed [37]. In recent years, GPU-based physics engines have become a new trend in large-scale simulation. For example, AirSim

and Genesis utilize GPU-accelerated physics engines that can efficiently handle complex 3D physical interactions and are suitable for large-scale multi-robot simulation [31,38]. However, these tools usually demand high-end computing hardware and require complex configuration and parameter optimization. ARGoS [25] is a simulator that can run quick large-scale simulations through a spatial partitioning parallel mechanism, dividing three-dimensional space into non-overlapping sub-regions assigned to independent physics engines. The system currently integrates four categories of physics engines (3D dynamic engines, 3D particle engines, 2D dynamic engines, and specialized engines), orchestrated by a unified entity state management interface. This architecture enables ARGoS to run rapid simulations of thousands of robots while fulfilling high-precision physical modeling requirements in specific scenarios [26].

Parallel computing is a key technology to support large-scale multi-robot simulation. Stage has high performance thanks to its lightweight 2.5D design, but it lacks support for parallel computing [37]. CoppeliaSim and MORSE's parallel computing capability mainly relies on the physics engine, leading to a marginal performance improvement [8,23]. Gazebo and Webots can benefit from both multi-thread and multi-device computation thanks to their integration with ROS 2 [17]. However, due to the constraints of their physics engine, parallel computing improvement remains limited and does not allow these simulators to scale to large numbers of robots. PyBullet supports multi-threading and GPU acceleration, which can efficiently handle large-scale physics computation, but it needs to rely on custom scripts for distributed simulation [7]. AirSim and Genesis excel in parallel computing, however, their configuration complexity and hardware requirements limit their wide adoption [31,38]. ARGoS has a significant advantage in multi-threaded and distributed computing as it can allocate simulation tasks to multiple CPU cores or computers, significantly improving simulation efficiency. ARGoS's event-driven mechanism further reduces unnecessary calculations, and its modular design and flexible configuration options make it easily adaptable to different speed and realism needs [27].

2.2 Simulated robotic platforms

ARGoS performs well in large-scale multi-robot simulation and, thanks to embedded cross-platform compilation, allows fast migration from simulation to real robots. However, such a high degree of consistency is mainly focused on a limited number of robotic platforms (e.g., Foot-bot, Kilobot, e-Puck, Thymio II, TurtleBot 3), resulting in the need to develop additional adaptation layers when migrating to other types of robots. Gazebo, Webots, and PyBullet, thanks to their ROS interface, leverage ROS's large community support and rich set of hardware drivers available to access a wide range of commercially available sensors and actuators [2,21]. At the same time, ROS provides a complete tool chain (rviz, rqt, rosbag, etc.) that significantly simplifies debugging, testing, monitoring, and data playback. Its standardized communication mechanisms for topics, services and actions also offer efficient communication and coordination between (potentially heterogeneous) robots. We believe that our ROS-2-ARGoS Bridge framework

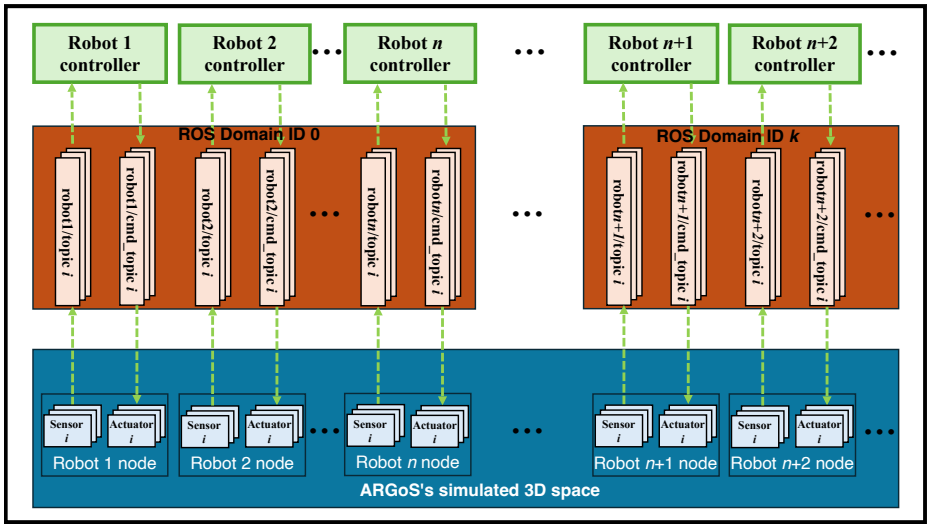

Fig. 2. An overview of the ROS-2-ARGoS Bridge framework. The ROS-2-ARGoS Bridge launches two ROS nodes for each simulated robot: one node is executed within ARGoS (blue rectangles on the bottom) and the other node is launched on the ROS side to execute the robot controller (green rectangles on the top). The two nodes of each robot exchange data through multiple ROS topics, one for each sensor and actuator that the robot is equipped with.

can foster the adoption of ARGoS as a simulator for large-scale robotics and, in turn, spawn the integration in ARGoS of several new robotic platforms for multi-robot and swarm robotics applications.

3 Methods

The ROS-2-ARGoS Bridge handles efficient bidirectional data flow between the ARGoS simulator and the ROS 2-based controllers of each robot through a modular architecture. We first give an overview of the framework, then we describe the temporal workflow of the simulation, and finally, we give more details on the most critical system components.

3.1 System Architecture

The overall architecture of the system is illustrated in Fig. 2. The ARGoS simulator (blue component at the bottom) updates the 3D simulated space comprising robots (with their sensors and actuators) and environmental entities (e.g., walls, lights, objects) with potentially time-varying characteristics (e.g., changes in light intensities). The ROS-2-ARGoS Bridge launches two ROS nodes (i.e., two processes) per simulated robot: one runs within ARGoS, while the other

executes the robot controller on the ROS side. The two nodes exchange data via the ROS's DDS (Data Distribution Service), more precisely, through dedicated ROS topics, one for each sensor and actuator that the robot is equipped with. Sensor topics are used to transfer sensor reading data from ARGoS to the robot controller, while actuator topics are used to transfer actuation control commands from the controller to ARGoS.

In the DDS, the primary mechanism for having different logical networks share a physical network is known as the ROS Domain ID. ROS 2 nodes within the same Domain ID can discover and communicate with each other, whereas ROS 2 nodes in different Domain IDs cannot. For each Domain ID, the DDS computes the UDP ports used for discovery and communication. Because each ROS 2 node uses two UDP ports, running several nodes in a single Domain ID may saturate the number of available ports on the computer. To avoid this limitation, robot nodes are distributed across multiple ROS Domain IDs. he assignment can be configured manually or automatically balanced by the framework across available Domain IDs. In our experiments, robots were automatically distributed with 50 robots per Domain ID, that is, (i.e., 100 nodes, each using two UDP ports). This strategy allows the system to scale up to large-scale simulations in the order of 10^3 robots, without requiring manual per-robot configuration.

Additionally, the ROS 2 DDS provides a distributed communication protocol, enabling the robot controllers to run on different computers or cluster nodes while interacting with the ARGoS simulator over the same ROS 2 network. Distributing the computation among different machines can potentially increase the simulation speed.

3.2 Temporal Workflow

The temporal workflow of a simulation is illustrated in Fig. 3, where we illustrate how various operations are executed by the three main system components (ARGoS, the Bridge, and ROS 2) and how data flows between them as time progresses (on the vertical axis, from top to bottom). The left column shows the typical sequence of phases of a simulation. The simulation is initialized (Init) by launching ARGoS and ROS 2, which loads the simulation environment and configures the ROS Domain, respectively. In the Setup phase, ARGoS starts the ROS-2-ARGoS Bridge, which is implemented as an ARGoS loop function. The bridge launches the robots' ROS nodes and configures the respective ROS topics.

The simulation then starts the Control Loop macro-phase, which repeats the two phases of Sensor Data Collection (SDC) and Control Command Execution and State Update (CCE&SU). During the SDC phase, ARGoS computes the simulated sensor readings, which are transmitted by the bridge to the ROS 2 controller via dedicated topics. The bridge also orchestrates time synchronization between ARGoS and ROS 2 by exposing the ARGoS simulation clock on a dedicated /clock topic and by gating the advancement of each simulation tick on the completion of the corresponding ROS callbacks. ROS 2 controllers, therefore, perceive the same notion of time as ARGoS: the simulator pauses after

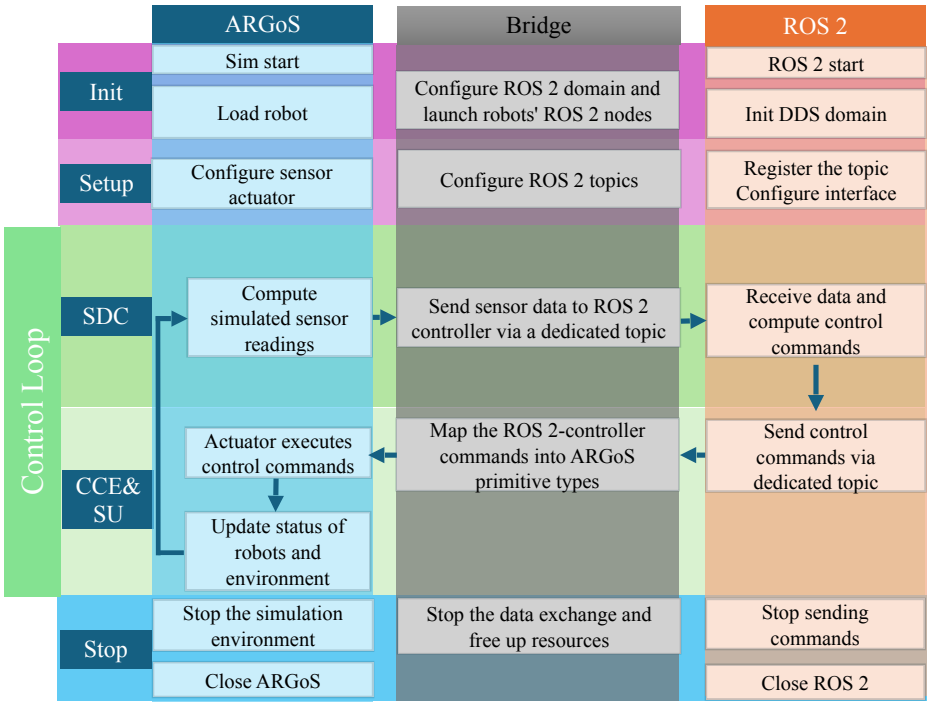

Fig. 3. Temporal workflow of a simulation where the ROS-2-ARGoS Bridge handles data exchange between the ARGoS simulator and the ROS 2 robot controller. Key phases include Init (initialization), Setup (configuration), SDC (sensor data collection, CCE&SU (control command execution and state update), and Stop (termination of the simulation).

providing sensor readings, waits for the controllers to process them and publish the resulting commands, and only then proceeds with the CCE&SU phase. Suppose a controller fails to respond before a configurable watchdog expires. In that case, the bridge records the violation, injects a safe fallback command (e.g., zero velocities, previous commands), and resumes the simulation to prevent the system from stalling. This mechanism guarantees deterministic replay, keeps ARGoS' clock authoritative even when ROS 2 nodes misbehave, and allows experiments to progress without manual intervention. In the CCE&SU phase, the bridge maps the received commands into ARGoS primitive types and forwards them to ARGoS for execution. Finally, ARGoS updates the state of the robots and the environment by executing the actuation commands (e.g., update robots' position) and any simulated temporal dynamics of the environment.

Eventually, when the termination condition is met (e.g., maximum simulation time is reached), ARGoS stops the simulation, the bridge halts data exchange, and ROS 2 shuts down (Stop phase).

3.3 Inter-robot Communication

In certain collective robotics application scenarios, especially in swarm robotics, it can be relevant to investigate the system dynamics when robots can only locally communicate. Such robots can only exchange messages with other robots (neighbors) located at a distance shorter than a given communication range. For example, robots may only be able to exchange messages by transmitting infrared signals to robots closer than 50 cm. Such local communication can improve the robotic system's scalability as saturation of the communication channel is prevented. However, because ROS 2 communication is handled through the DDS, robots (ROS nodes) in the same ROS Domain ID can always exchange data. Therefore, simulating distance-based inter-robot communication with traditional ROS-compatible simulators (e.g. Gazebo) can be complicated.

The ROS-2-ARGoS Bridge enables the seamless execution of ROS 2-based robot controllers with local inter-robot communication. During simulation, communication is handled analogously to sensors and actuators, exploiting distance-based neighbor selection. At each simulation step, ARGoS determines neighboring robots according to the communication device characteristics (e.g., Euclidean distance or line of sight), and copies the transmitted messages into the sensor data of those neighbors. The implementation of inter-robot communication mediated by ARGoS enables the simulation of both local and global communication, depending on application requirements and robot constraints.

4 Simulated Robotic Platforms

To run ROS-2-ARGoS simulations, the virtual model of the robot must be implemented in ARGoS. Each robot type is characterized by a specific set of sensors and actuators, which determine the external information accessible to the robot (e.g., environmental state) and how it changes its state (e.g., its positions) through physical interactions with the environment. The environment is simulated using a physics engine, and each robot's sensing and actuation must be modeled in the simulator. Like in any simulators, including realistic sensing and actuation noise requires dedicated tests to quantify such errors. Additionally, video visualization of the simulation requires a 3D model of the robot.

ARGoS already includes a bunch of models for popular robotics platforms, especially relevant for swarm robotics research. In addition to the Foot-bot [5], which is natively included in ARGoS, various plugins[4] allow simulating a variety of other popular robots, including Thymio II [28], Khepera IV [34], TurtleBot 3 [1], Kilobot [24], E-puck [20], and Crazyflie [35]. In our experiments, we used the TurtleBot 3 [1], which is a robotic platform widely used in several research laboratories studying collective robotics, and is supported in both ARGoS and Gazebo, allowing us to compare the performances of the two simulators.

Robot models that are implemented in ARGoS can be interfaced with ROS 2 through a robot-specific ROS-2-ARGoS Bridge. As shown in Fig. 2, the only exchanged information regards sensors and actuators. Hence, our framework can

[4] https://www.argos-sim.info/extensions.php

be extended by mapping the data exchanged between ARGoS and ROS 2. More specifically, ARGoS's sensor data map to ROS 2's controller sensor input, and ARGoS's actuator input parameters map to ROS-2 actuation output. In each simulation, each robot can be configured by including the set of sensors and actuators needed for the given application scenario. In our open-source code,[5] we implemented ROS-2-ARGoS Bridge for a few relevant sensors and actuators: colored blob perspective camera sensor, infrared (IR) proximity sensor, ground-color sensor, light sensor, differential wheels, LED, range-and-bearing local communication, and LiDAR. We configured them for the Foot-bot and TurtleBot 3 robots. These same sensors and actuators, with different parameters and noise levels, can also model other robotic platforms, e.g., the Thymio II robot [28].

5 Simulation Experiments

We test the performance of our simulation approach in a classical swarm robotics task: flocking [36] (see a simulation screenshot in Fig. 1). This scenario has been widely studied in the literature and can be used as a building block for various applications, such as environmental monitoring, distributed sensing, and collective transport [4,39]. To implement flocking, where a group of robots flock in a hexagonal lattice towards a light source placed in the task environment, we use a generalization of the Lennard-Jones potential parameterized with the same values of [26] (code available in the same repository[5]).

We evaluate the scalability of our proposed framework using simulated TurtleBot 3 swarms of size $N \in \{3, 5, 10, 20, 40, 80, 160, 320, 640, 1280\}$ performing flocking. On Linux systems, the ROS-2-ARGoS Bridge framework can theoretically support up to 7200 robots due to ROS 2 constraints.[6] Specifically, ROS 2 provides 120 Domain IDs, each supporting up to 120 processes; since each robot requires two processes, this limits the total number of robots. Here, we run our tests with up to 1280 robots as a proof of concept. For all experiments, we performed 10 independent trials, each consisting of 600 simulation time steps. With a step size of 0.1 s, each trial corresponds to one simulated minute. We measured the time taken to run the simulation, as well as CPU and memory usage. As ARGoS supports multi-threaded execution, we ran our experiments both with and without multi-threading enabled. Because the simulation were run on a 12-core machine, ARGoS was configured to use 12 threads.

In our experiments, ROS 2 nodes were initialized with 100 ROS 2 nodes per ROS Domain ID, comprising 50 ARGoS nodes and 50 controller nodes. This setting was adopted to avoid conflicts with ephemeral ports at large population sizes. We used the default Eclipse Cyclone DDS settings for all experiments.

All simulations were conducted on a computer running Linux Ubuntu 22.04.5 LTS, equipped with an AMD® Ryzen 9 7900 processor featuring 12 cores, each supporting 2 threads for a total of 24 threads, as well as 32 GB of RAM.

[5] https://github.com/CPS-Konstanz/argos3-ros2-bridge

[6] https://docs.ros.org/en/jazzy/Concepts/Intermediate/About-Domain-ID.html

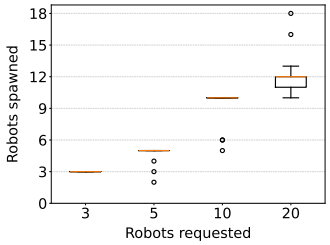

Fig. 4. Distribution of the number of robots successfully spawned in Gazebo across 100 simulation iterations for different swarm sizes. The orange line is the median, the box represents the interquartile range (IQR), the interval of the whiskers contains data points that are within 1.5 IQR, and circular markers indicate outliers.

We compare the performance of the proposed simulation framework with the most popular ROS 2-based simulator, Gazebo 11. We tested Gazebo's scalability by launching the simulator with an increasing number of TurtleBot 3 robots. We repeated each experiment 100 times and counted how many robots were successfully spawned after one minute. Figure 4 shows that Gazebo becomes unreliable beyond 3 robots and never succeeded in spawning 20 robots. Due to Gazebo's poor scalability performances, Gazebo experiments were restricted to the swarm sizes $N \in \{3, 5, 10\}$.

Figure 5 shows the average wall-clock time, CPU usage, and memory consumption for the flocking simulations with up to 1280 robots. The black dashed horizontal line denotes the real-time threshold (60 seconds), marking the point at which simulation time equals real-world time. Both Gazebo and our framework support simulation quicker than real-time (i.e., one simulated minute takes less than one real-world minute to be executed). However, for the same swarm sizes, Gazebo requires approximately eight times more execution time than our framework. ARGoS achieves faster-than-real-time performance for fewer than 30 robots (or 50 with multithreading). Overall, ARGoS's execution time scales approximately linearly with the swarm size N.

Gazebo uses more computation (linear increase) than our ARGoS framework, which shows sublinear scaling of CPU usage with increasing N. However, our framework uses more memory than Gazebo for the same swarm sizes. Interestingly, Gazebo's memory consumption decreased sharply for the largest swarm size (10 robots). On the other hand, our framework maintains a linear increase with N of the memory usage.

6 Conclusion

This work presents the ROS-2-ARGoS Bridge, a scalable framework for large-scale multi-robot simulation. By bridging the high-performance, multi-threaded ARGoS simulator with the widely adopted ROS 2 ecosystem, our framework enables efficient and scalable simulation of large robot swarms.

Our results demonstrate that, in terms of scalability, our framework largely outperforms the state-of-the-art simulator for ROS-based robots, Gazebo. The proposed framework is computationally efficient and scales linearly with increasing swarm sizes, allowing physics-based simulations with over a thousand robots.

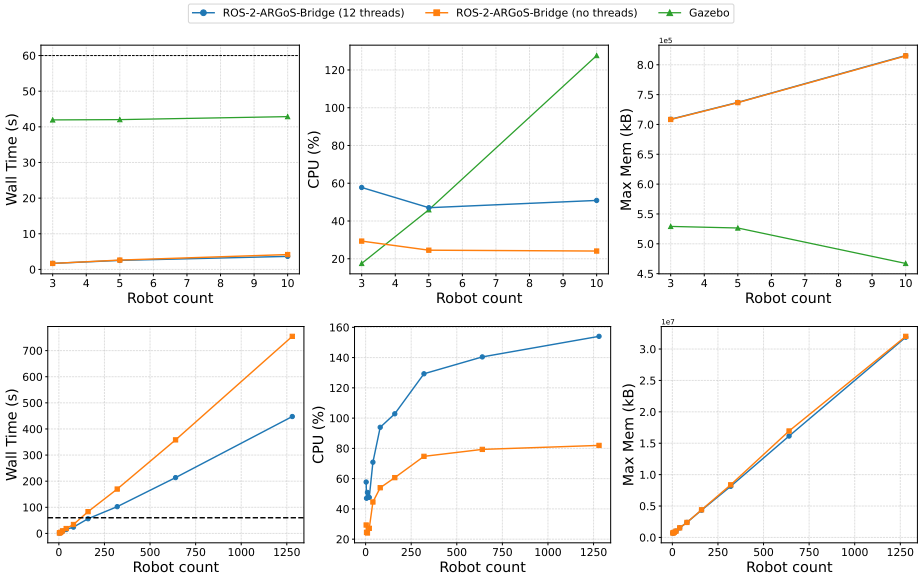

Fig. 5. Average wall clock time, CPU usage, and memory consumption for the ROS 2-based flocking simulations. (Top row) Performance comparison between Gazebo and the proposed simulation framework—ARGoS with ROS 2 Bridge with/without multi-threading. (Bottom row) Scalability performance of ARGoS in large-scale simulations.

We anticipate that the ROS-2-ARGoS Bridge will reduce development complexity, enhance code reusability, and streamline the transfer of swarm robotics algorithms from simulation to real-world applications. The proposed framework aims to accelerate the design, implementation, and testing of large-scale robotic systems beyond controlled lab environments, for example in automated warehouses and agricultural fields. Our approach supports the rapid expansion of real-world applications of swarm robotics.

Future work will investigate consolidating ROS components within a single process and leveraging intra-process communication to eliminate DDS-mediated UDP transport [22]. This approach is expected to reduce communication overhead and remove constraints such as limited UDP port availability, while preserving multi-threading capabilities and enabling distributed deployment via one ROS container per computer [15].

Acknowledgements This work has been partially supported by the DFG under Germany's Excellence Strategy – EXC 2117-422037984. The authors thank Carlo Pinciroli for helpful discussions and Raina Zakir for testing the software and improving the GitHub installation documentation.

Disclosure of Interests The authors have no competing interests to declare.

References

1. Amsters, R., Slaets, P.: Turtlebot 3 as a robotics education platform. In: International Conference on Robotics in Education (RiE). pp. 170–181. Springer (2019)
2. Andreiev, A., Sotnik, S.: Comparative analysis of robotics platform: Webots, Coppeliasim and Gazebo. Tech. rep., Repository of Kharkiv National University of Radio Electronics (2024)
3. Beni, G.: From swarm intelligence to swarm robotics. In: International Workshop on Swarm Robotics, LNCS, vol. 3342, pp. 1–9. Springer, Heidelberg (2004)
4. Berlinger, F., Gauci, M., Nagpal, R.: Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm. Science Robotics **6**(50), eabd8668 (2021)
5. Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., Mondada, F.: The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4187–4193. IEEE (2010)
6. Choi, H., Crump, C., Duriez, C., Elmquist, A., Hager, G., Han, D., Hearl, F., Hodgins, J., Jain, A., Leve, F., Li, C., Meier, F., Negrut, D., Righetti, L., Rodriguez, A., Tan, J., Trinkle, J.: On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. Proceedings of the National Academy of Sciences **118**(1), e1907856118 (2021)
7. Coumans, E., Bai, Y.: PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org (2016–2021)
8. Echeverria, G., Lassabe, N., Degroote, A., Lemaignan, S.: Modular open robots simulation engine: MORSE. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 46–51. IEEE (2011)
9. Gauci, M., Ortiz, M.E., Rubenstein, M., Nagpal, R.: Error cascades in collective behavior: A case study of the gradient algorithm on 1000 physical agents. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 1404–1412. IFAAMAS (2017)
10. Hamann, H.: Swarm robotics: A formal approach. Springer, Cham (2018)
11. Hamann, H., Reina, A.: Scalability in computing and robotics. IEEE Transactions on Computers **71**(6), 1453–1465 (2021)
12. Ikumapayi, O.M., Laseinde, O.T., Elewa, R.R., Ogedengbe, T.S., Akinlabi, E.T.: Swarm robotics in a sustainable warehouse automation: Opportunities, challenges and solutions. In: E3S Web of Conferences. vol. 552, p. 01080. EDP Sciences (2024)
13. Koenig, N., Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). vol. 3, pp. 2149–2154 vol.3 (2004)
14. Kuckling, J., Luckey, R., Avrutin, V., Vardy, A., Reina, A., Hamann, H.: Do we run large-scale multi-robot systems on the edge? More evidence for two-phase performance in system size scaling. In: 2024 IEEE International Conference on Robotics and Automation (ICRA). pp. 4562–4568. IEEE (2024)
15. Macenski, S., Soragna, A., Carroll, M., Ge, Z.: Impact of ROS 2 node composition in robotic systems. IEEE Robotics and Automation Letters **8**(7), 3996–4003 (2023)
16. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics **7**(66), eabm6074 (2022)

17. Mañas-Álvarez, F.J., Guinaldo, M., Dormido, R., Dormido-Canto, S.: Scalability of cyber-physical systems with real and virtual robots in ROS 2. Sensors **23**(13), 6073 (2023)
18. Maruyama, Y., Kato, S., Azumi, T.: Exploring the performance of ROS2. In: Proceedings of the 13th International Conference on Embedded Software (EMSOFT). pp. 1–10. IEEE (2016)
19. Michel, O.: Webots: Symbiosis between virtual and real mobile robots. In: Virtual Worlds: First International Conference (VW'98), LNCS, vol. 1434, pp. 254–263. Springer (1998)
20. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, vol. 1, pp. 59–65. IPCB: Instituto Politecnico de Castelo Branco, Portugal (2009)
21. Mower, C., Stouraitis, T., Moura, J.a., Rauch, C., Yan, L., Behabadi, N.Z., Gienger, M., Vercauteren, T., Bergeles, C., Vijayakumar, S.: ROS-PyBullet interface: A framework for reliable contact simulation and human-robot interaction. In: Proceedings of the 6th Conference on Robot Learning, PMLR, vol. 205, pp. 1411–1423. MLResearchPress (2023)
22. Naury, L., Gouguet, A., Lozenguez, G., Fabresse, L.: Communication isolation for multi-robot systems using ROS2. In: Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing. p. 850–858. SAC '25, ACM (2025)
23. Noori, F.M., Portugal, D., Rocha, R.P., Couceiro, M.S.: On 3D simulators for multi-robot systems in ROS: MORSE or Gazebo? In: 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR). pp. 19–24. IEEE (2017)
24. Pinciroli, C., Talamali, M.S., Reina, A., Marshall, J.A.R., Trianni, V.: Simulating Kilobots within ARGoS: models and experimental validation. In: Swarm Intelligence (ANTS 2018), LNCS, vol. 11172, pp. 176–187. Springer, Cham (2018)
25. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G.D., Ducatelle, F., Stirling, T., Gutiérrez, A., Gambardella, L.M., Dorigo, M.: ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5027–5034. IEEE (2011)
26. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., et al.: ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm intelligence **6**, 271–295 (2012)
27. Pitonakova, L., Giuliani, M., Pipe, A., Winfield, A.: Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators. In: Proceedings of the 19th Annual Conference Towards Autonomous Robotic Systems (TAROS), LNCS, vol. 10965, pp. 357–368. Springer (2018)
28. Riedo, F., Chevalier, M., Magnenat, S., Mondada, F.: Thymio II, a robot that grows wiser with children. In: 2013 IEEE Workshop on Advanced Robotics and its Social Impacts. pp. 187–193. IEEE (2013)
29. Rohmer, E., Singh, S.P., Freese, M.: V-REP: A versatile and scalable robot simulation framework. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1321–1326. IEEE (2013)
30. Rubenstein, M., Cornejo, A., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. Science **345**(6198), 795–799 (2014)

31. Shah, S., Dey, D., Lovett, C., Kapoor, A.: AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and Service Robotics, SPAR, vol. 5, pp. 621–635. Springer (2018)
32. Simoni, M.D., Kutanoglu, E., Claudel, C.G.: Optimization and analysis of a robot-assisted last mile delivery system. Transportation Research Part E: Logistics and Transportation Review **142**, 102049 (2020)
33. Smith, R.: Open dynamics engine (2006), https://ode.org/
34. Soares, J.M., Navarro, I., Martinoli, A.: The Khepera IV mobile robot: performance evaluation, sensory data and software toolbox. In: Robot 2015: Second Iberian Robotics Conference. AISC, vol. 417, pp. 767–781. Springer (2015)
35. Stolfi, D.H., Danoy, G.: An ARGoS plug-in for the Crazyflie drone. arXiv **2401.16948** (2024)
36. Turgut, A.E., Çelikkanat, H., Gökçe, F., Şahin, E.: Self-organized flocking in mobile robot swarms. Swarm Intelligence **2**, 97–120 (2008)
37. Vaughan, R.: Massively multi-robot simulation in Stage. Swarm intelligence **2**, 189–208 (2008)
38. Zhou, X., Qiao, Y., Xu, Z., Wang, T., Chen, Z., Zheng, J., Xiong, Z., Wang, Y., Zhang, M., Ma, P., et al.: Genesis: A generative and universal physics engine for robotics and beyond (2024), https://github.com/Genesis-Embodied-AI/Genesis
39. Zhou, X., Wen, X., Wang, Z., Gao, Y., Li, H., Wang, Q., Yang, T., Lu, H., Cao, Y., Xu, C., Gao, F.: Swarm of micro flying robots in the wild. Science Robotics **7**(66), eabm5954 (2022)