1. 机器人操作系统简介

机器人操作系统（ROS）是一种广受欢迎的开源机器人中间件框架，在学术界得到广泛的探索与研究。ROS并非完整的操作系统，其实质是一套用于构建和开发机器人应用程序的软件库和工具的集合体。从底层驱动程序到先进算法，再加上强大的开发工具，ROS几乎能够为机器人应用项目的开发提供一切软件层面的支持。由于具有节点对节点通信设计、支持多种编程语言（Python、Java、C++）、生态完善、集成度高、开源免费等特点，第一代机器人操作系统（ROS 1）自2007年问世以来就广受欢迎。ROS 1在各种各样的自主机器人、无人机以及智能汽车的研究和开发中得到广泛应用，例如，曾登上春晚的百度Apollo无人车、已在空间站工作的NASA机器人Robonaut 2等产品都是基于ROS 1进行设计和开发的[39]。

然而，随着ROS 1的逐步发展，其中存在的两大问题逐渐暴露出来：

第一个问题是通信实时性能有限。ROS 1的底层通讯机制主要依托于Linux系统进行通信，然而Linux系统并不能提供具有时间确定性的实时通信，这导致ROS 1通信的实时性能无法得到保证[40]。

第二个问题是系统可靠性不足，主要体现在单点故障上。ROS 1的Master节点是整个ROS网络的管理者和组织者，一旦Master节点出现宕机或者发生故障，整个ROS网络都将陷入无法通信的软件崩溃中。

这两个问题都根植于ROS 1的系统设计，导致ROS 1无法真正进入产业界，无法进行产品化，严重限制机器人操作系统的未来发展。

由于ROS 1存在通信实时性能有限、系统可靠性不足、仅支持Linux平台等难以解决的设计缺陷，ROS官方团队于2017年发布第二代机器人操作系统（ROS 2）。

经过全面且彻底的重新设计，ROS 2通过引入对象管理组织（OMG）主导的数据分发服务（DDS）进行通信，利用 DDS的丰富服务质量（QoS）来取代ROS 1中的通信机制，以改善和解决ROS 1中通信实时性能有限和系统可靠性不足等重大问题。DDS作为分布式系统中以数据为中心的发布-订阅通信标准，旨在实现设备和系统之间的实时、高吞吐量和低延迟的通信，并且内置对服务质量、安全性和可靠性的支持。通过引入DDS作为通信机制，ROS 2成功解决ROS 1中通信实时性能有限和系统可靠性不足等重大问题。此外，不同的机器人应用场景，所使用的控制平台可能存在巨大差异。为让所有的机器人应用程序都能迁移到ROS 2平台上，ROS 2可以跨平台部署在Linux、Windows、MacOS等操作系统上。目前，ROS 2广泛应用于机器人和自动化系统的开发，例如工业自动化、服务机器人、智能交通、医疗保健、农业机器人等。

1. 机器人操作系统相关工作

与机器人操作系统（ROS）相关的研究工作主要集中在提高ROS的实时性能上。例如，Wei[1]等人首次在多核处理器上提出一种名为RT-RTOS的实时ROS架构，RT-ROS 提供一种集成的实时/非实时任务执行环境，使得实时和非实时ROS节点分别运行在不同处理器内核的实时操作系统和Linux上。Saito[2]等人为ROS提出一个名为ROSCH 的实时调度框架，能够包含ROS本不支持的三个功能，分别是同步系统、基于固定优先级的有向无环图（DAG）调度框架以及故障安全功能。Suzuki[3]等人设计并实现一个可加载的内核模块框架，称为透明CPU/GPU协调机制上的实时ROS扩展，即ROSCH-G，用于在异构环境中调度ROS，从而无需修改操作系统内核和设备驱动程序。Saito[4]等人为ROS设计一种基于优先级的信息传输机制，以减少高优先级的ROS节点的执行时间和时间差异。不过，这些研究仅适用于早期版本的机器人操作系统，而且并未提供保证实时时序约束的分析方法。

自从2017年ROS 2发布以来，许多研究者针对ROS 2做出大量研究，大多数研究工作的目的在于评估和改进ROS 2的实时性能。例如，Blaß T[5]等人首次对 ROS 2 执行器进行正式的形式化建模和分析，对ROS 2应用程序的端到端延迟进行约束，为后续分析奠定基础。随后的工作，Casini[6]等人提出一种新颖的针对ROS 2任务链的响应时间分析，既考虑机器人工作负载中通常遇到的高执行时间方差，又考虑默认ROS 2回调调度程序的饥饿自由度。Yang[7]等人使用被称为回调组执行器的实时执行器替换ROS 2中的标准 RCLCPP执行器，并探讨与比较由此产生的实时性能。

ROS 2内部采用两种标准执行器：单线程执行器和多线程执行器。Tang[8]等人改进响应时间分析技术，通过形式建模对单线程执行器进行响应时间分析，并提出优化响应时间的优先级分配策略。Jiang[9]等人研究ROS 2中多线程执行器的实时调度和分析，并提出多线程执行器调度模型的形式化描述，从而在多线程执行器上开发任务链执行的响应时间分析技术。Choi [10]等人在最近的研究中，使用单线程执行器时，为ROS 2框架提出一种新颖的优先级驱动的任务链感知调度程序，并为所提出的调度程序提供端到端的延迟分析。此外，他们接着提出一个针对在ROS 2种多线程执行器上运行的任务链的综合响应时间分析框架[11]。近期，Liu[12]等人为ROS 2提出一种名为 RTeX 的新型无锁化的多线程执行器，以提高系统在运行时效率和时序可预测性方面的性能。

1. ROS 2执行器的评估测试

ROS 2支持单线程执行器与多线程执行器，当ROS 2系统中注册对应多条任务链的多个回调函数时，回调函数的优先级分配与回调函数组的线程绑定会对任务链的端到端延迟造成干扰，而任务链的端到端延迟是评估系统实时性能的重要衡量因素。本实验的目标在于评估与测试使用单线程执行器、多线程执行器以及多个单线程执行器时任务链端到端延迟的不同，从而研究与分析不同类型执行器对于系统实时性能的影响。

本实验在ROS 2系统中注册三条任务链，每条任务链中由三个回调函数组成，如图所示，其中，三条任务链中的九个回调函数分别编号1至9，并且每条任务链的第一个回调函数是由时间触发的定时器，其余两个回调函数是由事件触发的订阅者。



在下列六组配置的情况下，本实验测试三条任务链的端到端延迟的变化情况：

1. 九个回调函数全部加入一个单线程执行器中（SingleThread）；回调顺序是{1,2,3,4,5,6,7,8,9}。
2. 九个回调函数全部加入一个多线程执行器中，并且多线程执行器的线程数目为三（MultiThread）；回调顺序是{1,2,3,4,5,6,7,8,9}。
3. 将九个回调函数分成三组，回调顺序分别是{1,2,3}、{4,5,6}与{7,8,9}，每个回调函数组加入一个单线程执行器中（MultiExecutor1）；
4. 将九个回调函数分成三组，回调顺序分别是{1,5,9}、{2,6,7}与{3,4,8}，每个回调函数组加入一个单线程执行器中（MultiExecutor2）；
5. 将九个回调函数分成三组，回调顺序分别是{1,4,7}、{2,5,8}与{3,6,9}，每个回调函数组加入一个单线程执行器中（MultiExecutor3）；
6. 将九个回调函数分成三组，分别是{1,4,7}、{8,5,2}与{9,6,3}，每个回调函数组加入一个单线程执行器中（MultiExecutor4）。

本实验是在一台服务器电脑上进行，该桌面PC配备 16 核CPU，运行频率固定为 1.20GHz。每个 CPU 有 16.5MB 缓存，系统配备 62.5GB DRAM。本实验使用 ROS 2 Humble 版本，并采用了默认服务质量（QoS）设置的进程内 API。每个定时器回调函数的执行周期设置成300ms，每个回调函数的执行时间设置成100ms，消息大小设置为 4 字节，具体为 Int32，以确保将消息传输的开销降至最低。每个实验的运行时间超过五分钟，为考虑初始化影响，前500个样本被丢弃。在不同的配置下，每条任务链的端到端延迟的变化情况如图所示：

图表

描述已自动生成

在不同的实验配置下，每条任务链的端到端延迟的更详细细节如下表所示,，其中，CPU利用率表示在ROS 2系统运行过程中桌面PC的每秒CPU平均利用率，Switch表示在ROS 2系统运行过程中工作线程在用户态与内核态之间的每秒平均切换次数。

pidstat -u process\_name -u -w 1

每1秒钟获取一次two\_node\_pipeline线程的切换次数和CPU利用率。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Single-Threaded | Multi-Threaded | Multi-Executor1 | Multi-Executor2 | Multi-Executor3 | Multi-Executor4 |
| CPU利用率 | 46% | 45% | 38% | 41% | 44% | 45.50% |
| Switch | 64.85 | 45.8 | 0 | 0 | 0 | 0 |
| 第一条链平均值  （ms） | 41.76 | 40.27 | 38.65 | 40.05 | 41.1 | 40.08 |
| 第一条链STD  （ms） | 3.87 | 6.64 | 6.34 | 8.35 | 6.57 | 5.62 |
| 第二条链平均值  （ms） | 41.91 | 40.33 | 38.41 | 38.7 | 39.88 | 44.79 |
| 第二条链STD  （ms） | 3.99 | 6.71 | 5.95 | 3.76 | 4.55 | 4.19 |
| 第三条链平均值  （ms） | 41.96 | 40.68 | 38.34 | 39.58 | 40.85 | 44.61 |
| 第三条链STD  （ms） | 4.19 | 5.77 | 5.8 | 5.9 | 6.37 | 4.41 |

实验结果显示，使用不同类型的执行器，或者让具有不同优先级的回调函数注册在不同的执行器上运行，对任务链的端到端延迟存在显著影响。

参考文献

1. Wei H, Shao Z, Huang Z, et al. RT-ROS: A real-time ROS architecture on multi-core processors[J]. Future Generation Computer Systems, 2016, 56: 171-178.
2. Saito Y, Sato F, Azumi T, et al. Rosch: real-time scheduling framework for ros[C]//2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). IEEE, 2018: 52-58.
3. Suzuki Y, Azumi T, Kato S, et al. Real-time ros extension on transparent cpu/gpu coordination mechanism[C]//2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC). IEEE, 2018: 184-192.
4. Saito Y, Azumi T, Kato S, et al. Priority and synchronization support for ROS[C]//2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA). IEEE, 2016: 77-82.
5. Blaß T, Casini D, Bozhko S, et al. A ROS 2 response-time analysis exploiting starvation freedom and execution-time variance[C]//2021 IEEE Real-Time Systems Symposium (RTSS). IEEE, 2021: 41-53.
6. Casini D, Blaß T, Lütkebohle I, et al. Response-time analysis of ROS 2 processing chains under reservation-based scheduling[C]//31st Euromicro Conference on Real-Time Systems. Schloss Dagstuhl, 2019: 1-23.
7. Yang Y, Azumi T. Exploring real-time executor on ros 2[C]//2020 IEEE international conference on embedded software and systems (ICESS). IEEE, 2020: 1-8.
8. Tang Y, Feng Z, Guan N, et al. Response time analysis and priority assignment of processing chains on ROS2 executors[C]//2020 IEEE Real-Time Systems Symposium (RTSS). IEEE, 2020: 231-243.
9. Jiang X, Ji D, Guan N, et al. Real-time scheduling and analysis of processing chains on multi-threaded executor in ros 2[C]//2022 IEEE Real-Time Systems Symposium (RTSS). IEEE, 2022: 27-39.
10. Choi H, Xiang Y, Kim H. PiCAS: New design of priority-driven chain-aware scheduling for ROS2[C]//2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE, 2021: 251-263.
11. Sobhani H, Choi H, Kim H. Timing Analysis and Priority-driven Enhancements of ROS 2 Multi-threaded Executors[C]//2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE, 2023: 106-118.
12. Liu S, Jiang X, Guan N, et al. RTeX: an Efficient and Timing-Predictable Multi-threaded Executor for ROS 2[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2024.