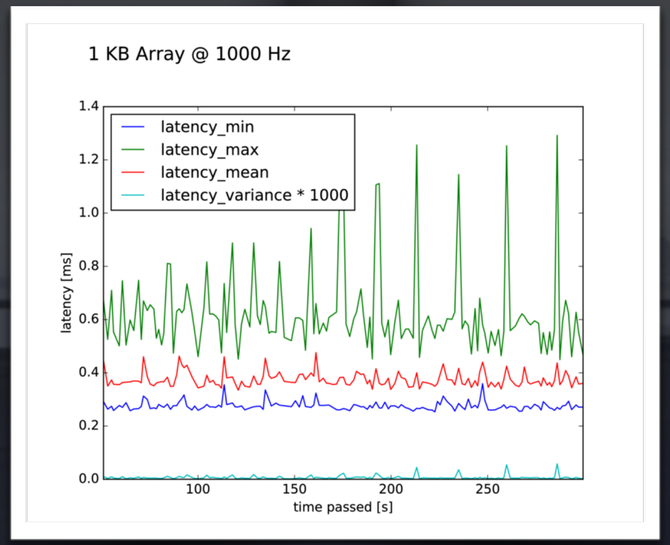
Middleware Performance Testing

Andreas Pasternak
Dejan Pangercic

Apex.AI®

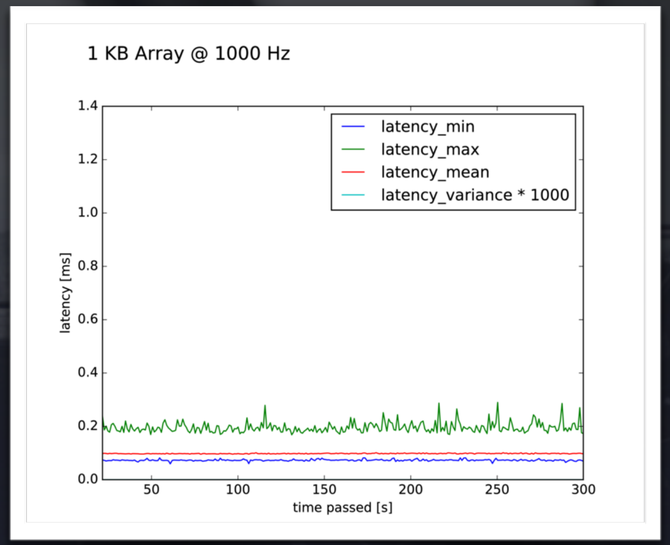
Goal: Benchmark middleware and optimize configurations

Example: Two latency graphs created on a NVIDIA Drive PX2 measuring ROS 2 latency before and after DDS and RMW optimization.



Before DDS and RMW improvements

1. Async ASIO Networking
2. Linux default sysctl settings



After DDS and RMW improvements

1. Synchronous UDP reception: github.com/eProsima/Fast-RTPS/commit/d2c2c8e536fd15eeadcf46a228b10f0d37a9648d
2. Tuned sysctl settings:

net.ipv4.udp_mem="102400 873800 16777216"
net.core.netdev_max_backlog="30000"
net.core.rmem_max="20971520"
net.core.wmem_max="20971520"
net.core.rmem_default="20971520"
net.core.wmem_default="20971520"

Introducing Performance Tests

Purpose

- To benchmark communication middleware

Implementation

- Plugin-based system to support additional middleware
- Separates middleware operations and measurements
- Supports inter- and intra-process and inter-machine benchmarks
- Real time aware software architecture
- Measures over 20 metrics, such as the distribution of latency
- Logging system allows exact matching of configuration and results
- Scripts to visualize log files and run large batches of experiments
- Could be integrated as part of the CI and testing framework

Open Source

https://github.com/ApexAI/performance_test

DDS Quality of Service

Quality of Service (QoS) is a contract between publisher and subscriber

Examples

- Lost data should be retransmitted or not
- Messages should be sent with a certain frequency or not have more than a certain delay
- Sets of samples should arrive as transactional entity
- Data should not be transmitted to a subscriber more often than a certain limit to not overload it
- History of data should be kept

QoS settings can have significant impact on performance

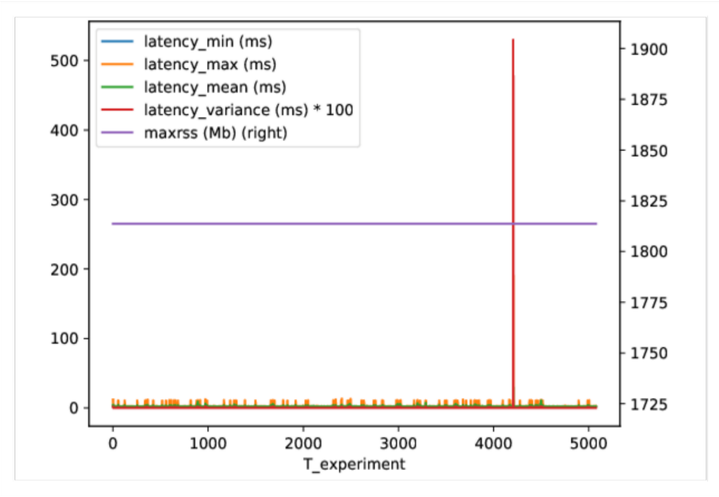
Apex.AI performance test allows to automatically benchmark all QoS permutations

	QoS Policy
Data Availability	**DURABILITY**
	DURABILITY SERVICE
	LIFESPAN
	HISTORY
Data Delivery	**PRESENTATION**
	RELIABILITY
	PARTITION
	DESTINATION ORDER
	OWNERSHIP
	OWNERSHIP STRENGTH
Data Timeliness	**DEADLINE**
	LATENCY BUDGET
	TRANSPORT PRIORITY
Resources	**TIME BASED FILTER**
	RESOURCE LIMITS
Configuration	**USER_DATA**
	TOPIC_DATA
	GROUP_DATA

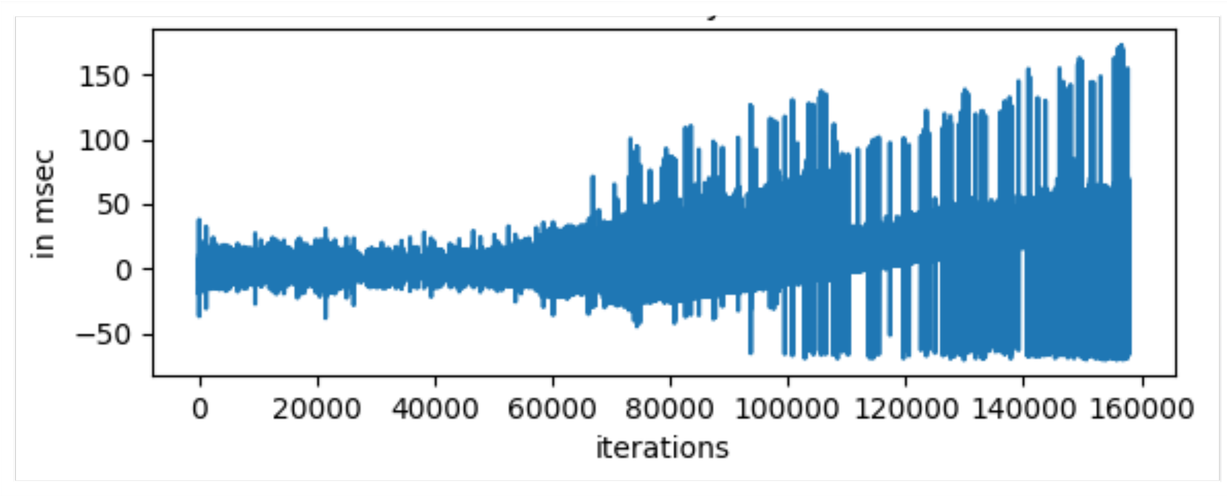
Hakiri, Akram & Berthou, Pascal & Gayraud, Thierry. (2010). Addressing the Challenge of Distributed Interactive Simulation With Data Distribution Service. **https://arxiv.org/pdf/1008.3759.pdf**

Common Problems

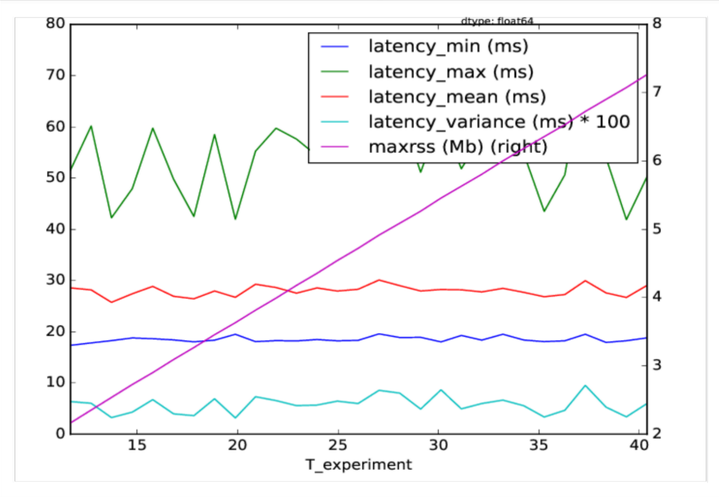
Rare latency spikes



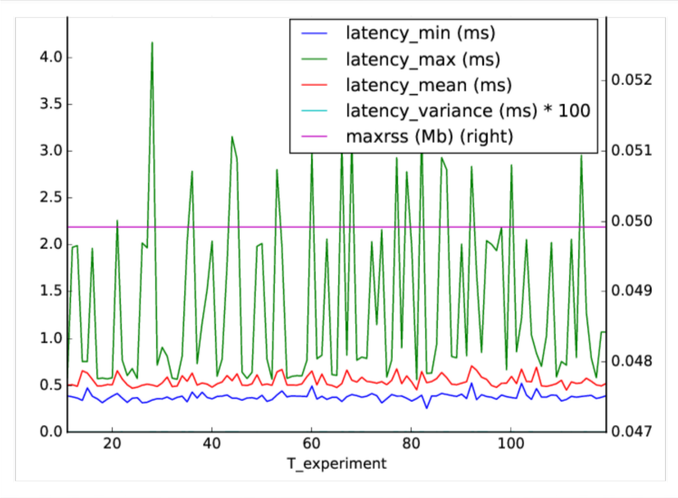
Increased latency over time



Memory leaks



Jitter



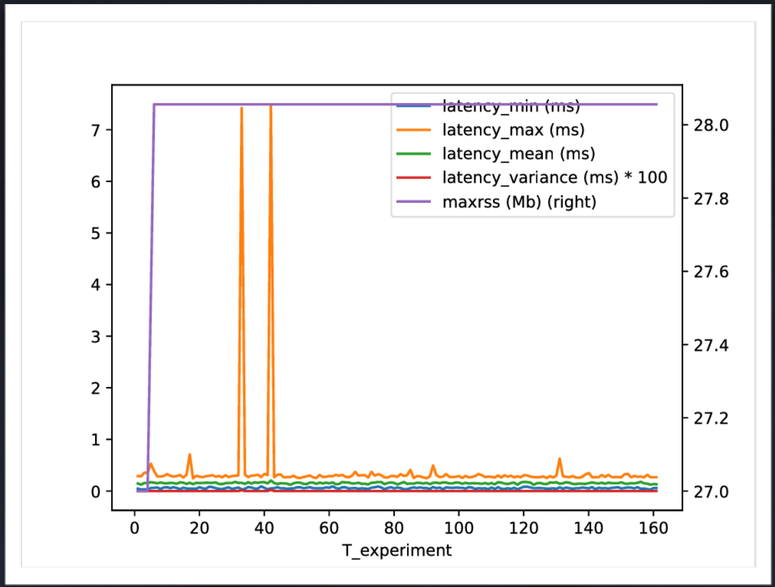
Simple Experiment Running

$ ros2 run performance_test perf_test -c ROS2 -t Array1k -l log
$ less log*

Experiment id: 5a005310-958f-41ec-9560-b2fa7cb707b8
Logfile name: log_Array1k_24-08-2018_14-09-38
Communication mean: ROS2
DDS domain id: 0
QOS: Reliability: BEST_EFFORT Durability: VOLATILE History kind: KEEP_ALL History depth: 1000 Sync. pub/sub: 0
...
---EXPERIMENT-START---

received,	sent,	lost,	relative_loss,	data_received,	latency_min,	latency_max,	latency_mean,	latency_variance
889,	889,	4,	0.00,	925119,	0.03156,	0.2739,	0.144,	2.096e-06,
880,	881,	0,	0.00,	916003,	0.04648,	0.3809,	0.1432,	2.412e-06,
888,	887,	0,	0.00,	923955,	0.04162,	0.3489,	0.1413,	1.859e-06,
886,	887,	0,	0.00,	922168,	0.05146,	0.4387,	0.1449,	1.71e-06,
899,	898,	0,	0.00,	934995,	0.06674,	0.4733,	0.1432,	1.495e-06,

python3 src/performance_test/performance_test/helper_scripts/performance_test_file_reader.py.



There are a lot of configuration options

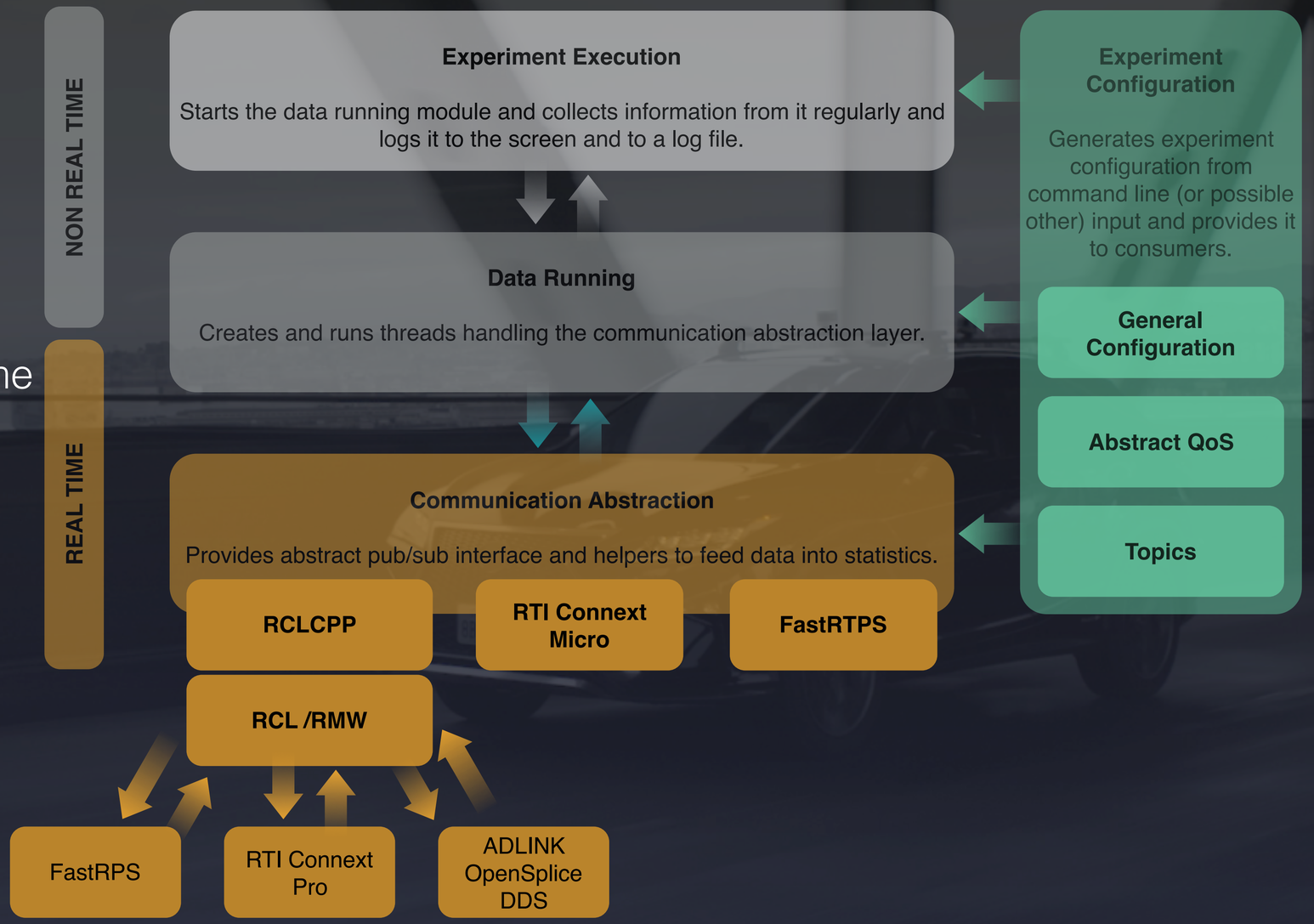
$ ros2 run performance_test perf_test --help
Allowed options:

-h [--help]	Print usage message.
-l [--logfile] arg	Optionally specify a log file.
-r [--rate] arg (=1000)	The rate data should be published. Defaults to 1000 Hz. 0 means publish as fast as possible.
-c [--communication] arg	Communication plugin to use (ROS2, FastRTPS, ConnextDDSMicro)
-t [--topic] arg	Topic to use. Use --topic_list to get a list.
--topic_list	Prints list of available topics and exits.
--dds_domain_id arg (=0)	Sets the DDS domain id.
--reliable	Enable reliable QOS. Default is best effort.
--transient	Enable transient QOS. Default is volatile.
--keep_last	Enable keep last QOS. Default is keep all.
--history_depth arg (=1000)	Set history depth QOS. Defaults to 1000.
--disable_async.	Disables asyc. pub/sub.
--max_runtime arg (=0)	Maximum number of seconds to run before exiting. Default (0) is to run forever.
-p [--num_pub_threads] arg (=1)	Maximum number of publisher threads.
-s [--num_sub_threads] arg (=1)	Maximum number of subscriber threads.
--use_ros_shm	Use Ros SHM support.
--use_drive_px_rt	Enable RT. Only the Drive PX has the right configuration to support this.
--use_single_participant	Uses only one participant per process. By default every thread has its own.
--no_waitset	Disables the wait set for new data. The subscriber takes as fast as possible.
--no_micro_intra	Disables the Connext DDS Micro INTRA transport.

Architecture

Architectured from ground up for extensibility and flexibility:

- Separates data logging, data aggregation, and data collection
- Designed to be extended with support for various middleware
- Separates real time and non real time components
- Supports arbitrary number of publisher and subscription threads



Advanced Features

New middleware can be added in 3 simple steps:

1. Add middleware types to the code generation
2. Add middleware types to the topics
3. Add middleware plugin:
    1. Add QoS mappings (if supported)
    2. Implement plugin API: publish(..) / update_subscription()

Helps you find and fix memory allocations:

```
$ cd perf_test_ws/src
$ git clone https://github.com/osrf/osrf_testing_tools_cpp.git
$ cd .. && volvon build --cmake-args -DCMAKE_BUILD_TYPE=Release
$ export LD_PRELOAD=$(pwd)/install/lib/libmemory_tools_interpose.so
$ ros2 run performance_test perf_test  -c ROS2 -t Array1k -l log --memory_check

Stack trace (most recent call last):
#7    Object "/home/andreas.pasternak/ros2_ws/install/lib/libfastrtps.so.1", at 0x7f9e6673824f, in
std::__cxx11::list<eprosima::fastrtps::rtps::RTPSWriter*, std::allocator<eprosima::fastrtps::rtps::RTPSWriter*> >::~list()
```

Conclusion

Performance problems can be hard to track down

Open source solution:
https://github.com/ApexAI/performance_test

- Helped **Apex.AI** and DDS providers find and fix a lot of bugs and performance issues
- Supports ROS 2, Connext DDS Micro, and FastRTPS (directly)
- Already used by various DDS providers

Apex.AI welcomes middleware implementers to integrate their middleware