

SEP 2 - 1D & 2D Experiment channel enhancement

Tiago Coutinho

04/01/2009

Contents

Introduction

This document describes a sardana enhancement proposal (SEP) for the 1D and 2D experiment channels. It is an extension of [experiment management] and [1D experiment channel].

Motivation

1D and 2D data are an important part of the data acquisition process. Because of the amount of data involved and due to the fact that the several sardana components (HW, Pool, MacroServer, Clients) can be distributed over several machines it becomes imperative to find a solution that minimizes data transfer without compromising flexibility.

It is therefore important to have the following points in mind when designing the system to support this kind of demands:

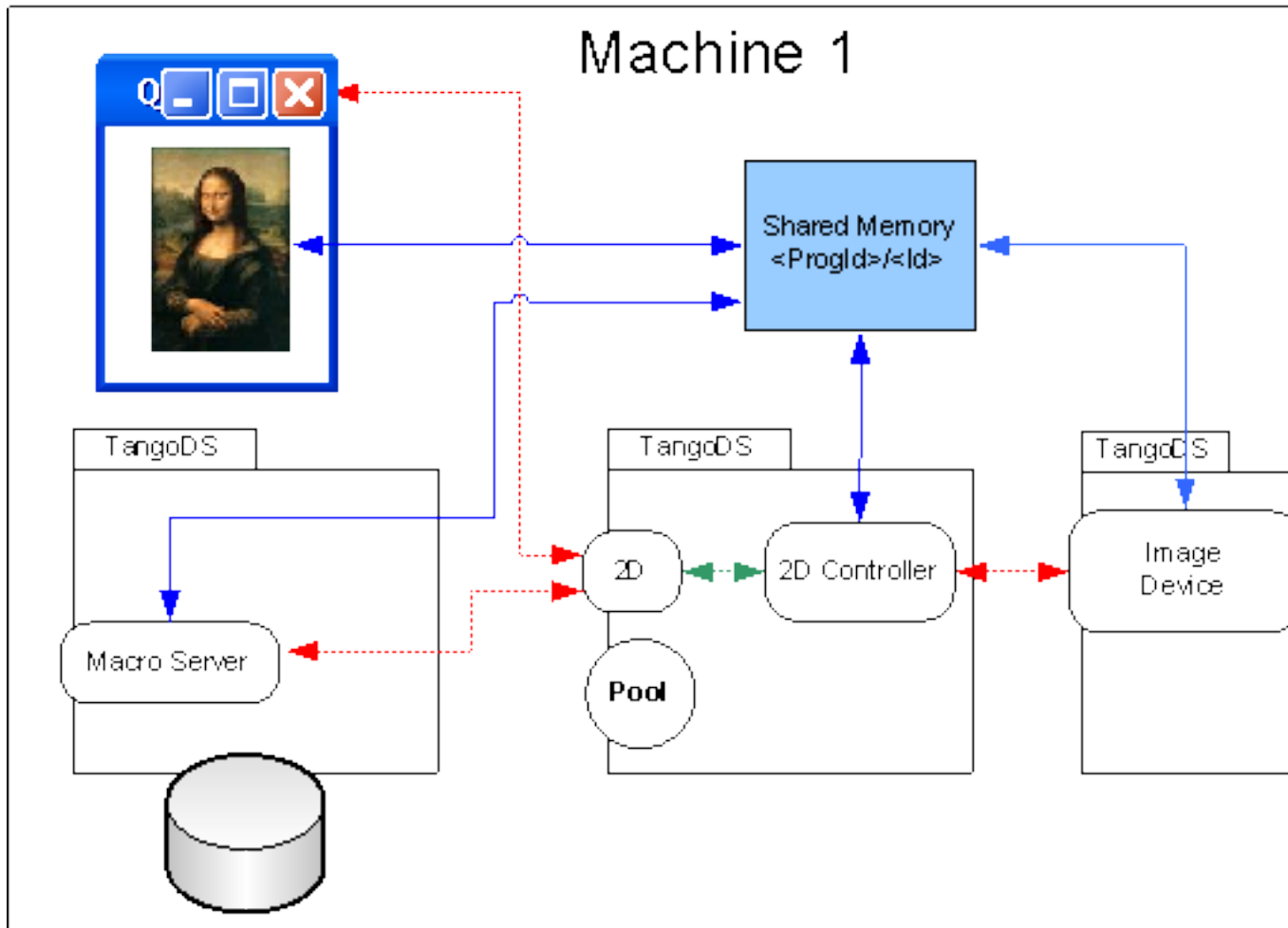
- minimize network data transfers
- minimize interprocess data transfers over the transport layer (e.x.: tango channel with CORBA interprocess optimizations below)
- minimize intraprocess data copy
- optimize data storage
- provide user with interactive feedback

Use cases

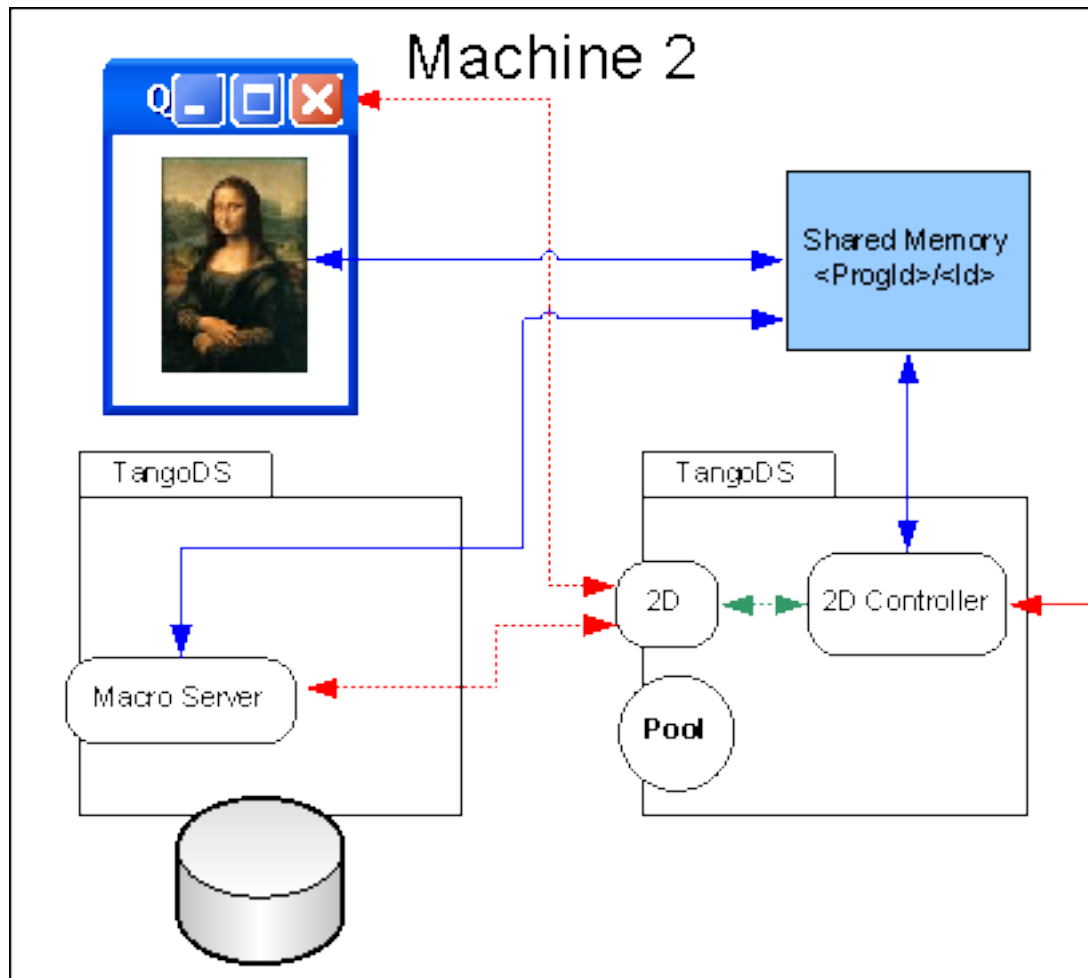
This chapter tries to give some examples of possible scenarios. The several use cases provided are focused on the different possible architectures that can be mounted on a real system. The examples provided focus on 2D data but the same principle applies also for 1D data. The implementation details like shared memory library, tango interface of the several devices or controller API will be discussed later.

The scenario (scenario 1) that provides less data transfer over the network comprises a single machine running all the device server daemon processes, as well as the final end user GUI application. The first figure shows a graphical representation of this scenario.

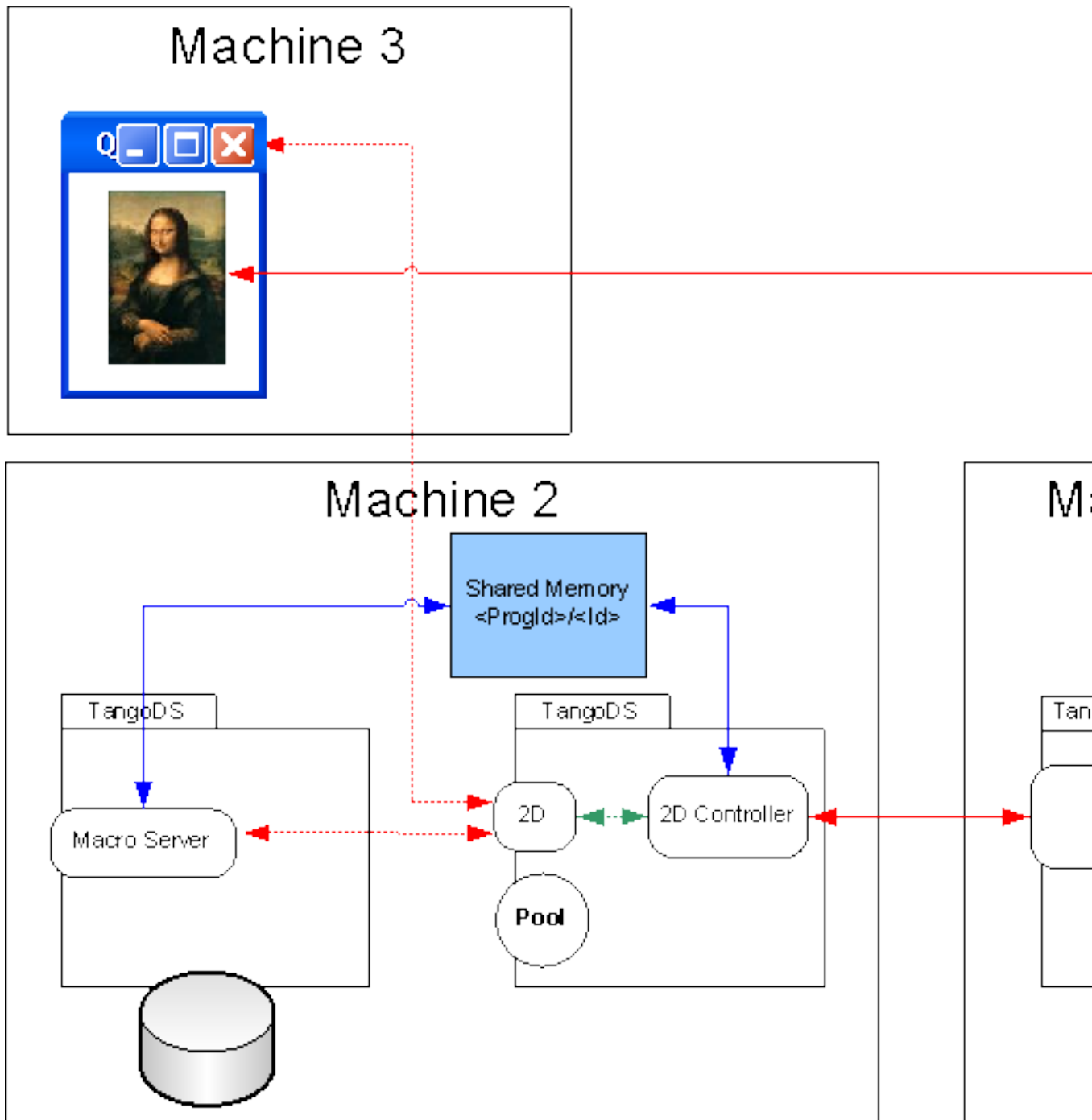
Figure 1: Scenario 1



Scenario 1



Scenario 2



Scenario 3

Considering this first scenario, because all processes run on the same machine, the system should be able to profit as much as possible from the fastest interprocess communication available: shared memory. The Tango channels will be used to transfer configuration information (like data

location, ROI).

For a given configuration there will be a shared memory owner (the writer) and several shared memory readers. The writer depends on the capabilities of each of the device servers.

There can be more than one shared memory but only one of the processes should be the writer.

Image device server as shared memory writer

If the image device server that reads directly from the HW has the capacity to store the data in shared memory, this should be the preferred writer for the data. The Pool, Macro Server and GUIs become the readers of this memory. With this configuration the data transfers are minimized as everything flows through the a single shared memory accessible to all processes.

Although very efficient, this is by far not the most common case because all of the image device servers that are implemented up to this date don't use shared memory. Also many times the image device server runs on an independent machine due to HW restrictions (e.x.: HW driver demands a specific OS) or to design options. In this case the shared memory is useless because all other processes run in different machines (scenarios 2 and 3).

Pool device server as shared memory writer

In this configuration, the 2D controller in the Pool server is responsible for writing the data into shared memory. The data can be fetched from many different ways. It depends on the person who implements the 2D controller for a specific HW to decide the best way to do this. The most common ways to do this are getting the data from an image device server through a tango channel or accessing directly to the HW driver.

In this scenario, the shared memory can be accessed for reading by the macro server or by the end user GUI.

MacroServer server as shared memory writer

The macro server should become the shared memory writer if the either the pool is running on a different machine or if the macro server needs to process the raw image in some way before showing it in the GUI and/or storing it.

Conclusions

Scenario 1 is only a reality for small laboratories where there are few HW elements to be controlled allows for a single computer to contain all necessary software. This is often not the case in a synchrotron beamline where the complexity of the control system forces the software to be distributed among several machines.

Details

Some key decisions have to be made regarding the concrete Sardana implementation in order to be able to obey the previous use cases.

Shared memory

The shared memory library is one of the key decisions. Some possibilities are:

1. SPS - SPEC shared memory
 - (a) Advantages:
 - i. no global lock over the shared memory
 - ii. simple API
 - iii. available in C and python
 - iv. developers know how
 - v. compatible with existing GUI applications (newplot, PyMCA)
 - (b) Disadvantages:
 - i. only works on linux
 - ii. limited data types
2. QSharedMemory - shared memory from Qt library
 - (a) Advantages:
 - i. platform independent
 - ii. available in C++ and python
 - (b) Disadvantages:
 - i. there is a global lock over the entire shared memory
 - ii. dependency on the huge Qt library
3. boost interprocess - shared memory library from boost
 - (a) Advantages:
 - i. platform independent
 - (b) Disadvantages:
 - i. there is a global lock over the entire shared memory
 - ii. only available in C++
 - iii. complex API (template based C++)

TwoDExpChannel tango device

This section describes the Tango interface for the TwoDExpChannel device that is exported in the Pool device server.

You may have noticed the usage of the DevEncoded type for the Value attribute.

References

- [experiment management] T. Emmanuel, Dec 2006, "Sardana" device pool experiment management
- [1D experiment channel] N. Maria-Teresa, July 2008, One dimensional experiment channel in Sardana