参数服务器的应用场景

相当于一个独立容器，　允许不同的节点 向这个容器 访问 或 存储数据

假设我们有 n 个节点，参数服务器 是独立于这些节点

n 个节点向参数服务器 存储数据　向 参数服务器 拿数据

全局路径规划　和局部路径规划 都会使用 无人车的尺寸 在规划车的轨迹的时候

以共享的方式 实现不同节点之间数据的交换

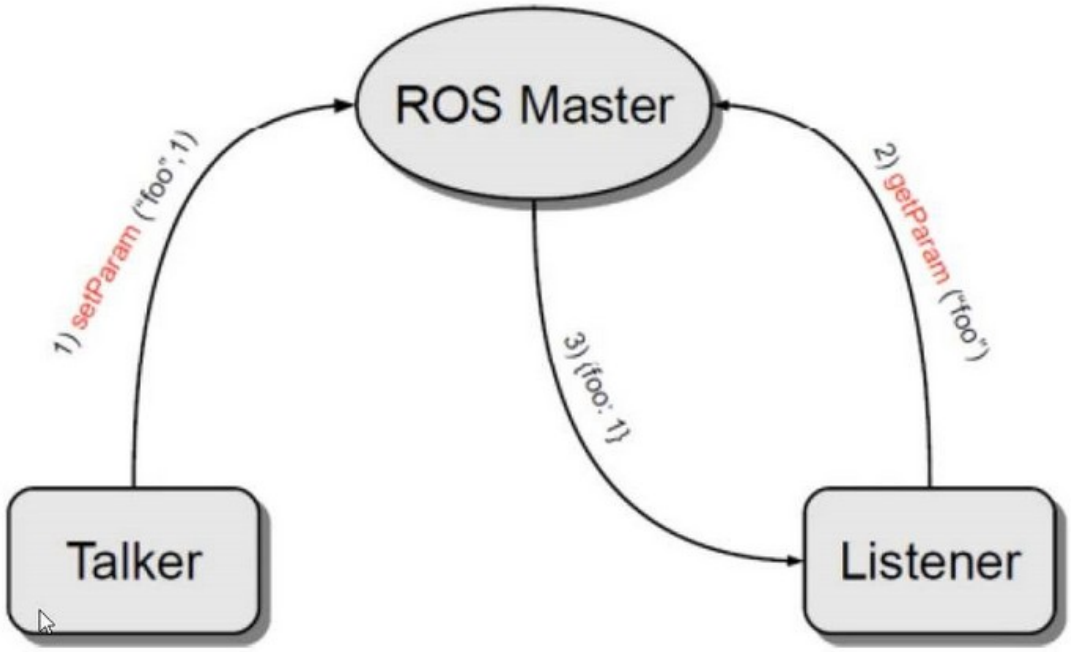
存储一些节点的共享数据，类似于全局变量

实现参数的增加 删减等操作

参数服务器的理论模型如下图所示
Master： 管理者 公共容器的保存参数 将参数保存到参数列表
talker： 向容器设置参数。向容器提交参数,
listener： 从容器获取参数 向管理者发出请求

- ROS Master (管理者)
- Talker (参数设置者)
- Listener (参数调用者)

ROS Master 作为一个公共容器保存参数，Talker 可以向容器中设置参数，Listener 可以获取参数。



1、talkeer 向参数服务器 master 提交参数
2、master 会将参数保存到参数服务器中，参数服务器 会有个 list
3、listener 回想 master 发送请求回去参数
4、master 向 listener 发送参数

参数服务器的通信方式 使用 RPC 地址

RPC 地址 是非高性能的，最好是存储简单数据类型 静态的非 2 进制数据

32bit integers 4 个字节的 整形数
doubles
strings
bool
iso8601 dates 时间类型的数据 iso8601 约束的时间格式存储数据
list 列表
字典 由键值队实现
base64 的二进制数据

实现参数服务器的 增 删 改 查

通过两套 api 实现
ros::NodeHandle
ros::param 实现
注意初始化的节点实现
注意 setParam 和 param 两个方式
键值重复恢复 这里面的数值 就会发送变化
key 的 value 这里是 string 类型 我们想要 double 和 string 类型的 value

```cpp
#include"ros/ros.h"



/*
实现 参数的新增和修改
ros::NodeHandle
setParam()
ros::param
set()
都有这个setParam的方法
配置
add_executable
target_link_libraries
*/

int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "");
    ros::init(argc,argv,"serverparams11111");
    ros::NodeHandle nh;
    /*参数新增 ------------------------*/
    //nh 方案
    nh.setParam("type","xiaohuangche");
    nh.setParam("radius",0.12);//这里设置的参数类型是 float类型 最开始我设置成了 "“ string 类型的数据  所有无法返回 float类型的东西
    //ros::param
    //传入一个键 一个值 键要独立的
    //键值重复会覆盖
    ros::param::set("type_param", "xiaobai");
    ros::param::set("radius_param",0.15);

    ROS_INFO("参数设置成功");
    //已经存在的健 再次设置就算修改
    ros::param::set("radius_param",0.1211);


    return 0;
}
```

在响应 demo03_ws 的功能包空间下 通过 rosparam list 可以显示 set 的 param 之后 rosparam get /键值 获得对应的 value

```
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ source ./devel/setup.bash
```

编译的配置文件
只有 add_executable
target_link_libraries
主需要修改上述两个文件 就可以完成文件的配置

```
136  add_executable(demo01_param_set src/demo01_param_set.cpp)
137  add_executable(demo02_param_get src/demo02_param_get.cpp)
138  ## Rename C++ executable without prefix
139  ## The above recommended prefix causes long target names, the following renames the
140  ## target back to the shorter version for ease of user use
141  ## e.g. "rosrun someones_pkg node" instead of "rosrun someones_pkg someones_pkg_node"
142  # set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node PREFIX "")
143
144  ## Add cmake target dependencies of the executable
145  ## same as for the library above
146  # add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPOR
147
148  ## Specify libraries to link a library or executable target against
149  target_link_libraries(demo01_param_set
150    ${catkin_LIBRARIES}
151  )
152  target_link_libraries(demo02_param_get
153    ${catkin_LIBRARIES}
154  )
155
```

通过 rosparam 的 从终端 terminal 中看我们的设置

```
问题 2  输出  调试控制台  终端  端口  注释
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ source ./devel/setup.bash
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosrun plumbing_param_server demo01_param_set
[ INFO] [1710574250.277586912]: 参数设置成功
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosparam list
/radius
/radius_param
/rosdistro
/roslaunch/uris/host_qinghuan_system_product_name__42643
/rosversion
/run_id
/type
/type_param
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosparam get /radius_param
'0.151'
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$
```

```
问题 2  输出  调试控制台  终端  端口  注释
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosparam list
/radius
/radius_param
/rosdistro
/roslaunch/uris/host_qinghuan_system_product_name__42643
/rosversion
/run_id
/type
/type_param
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosparam get /type_param
xiaobai
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$
```

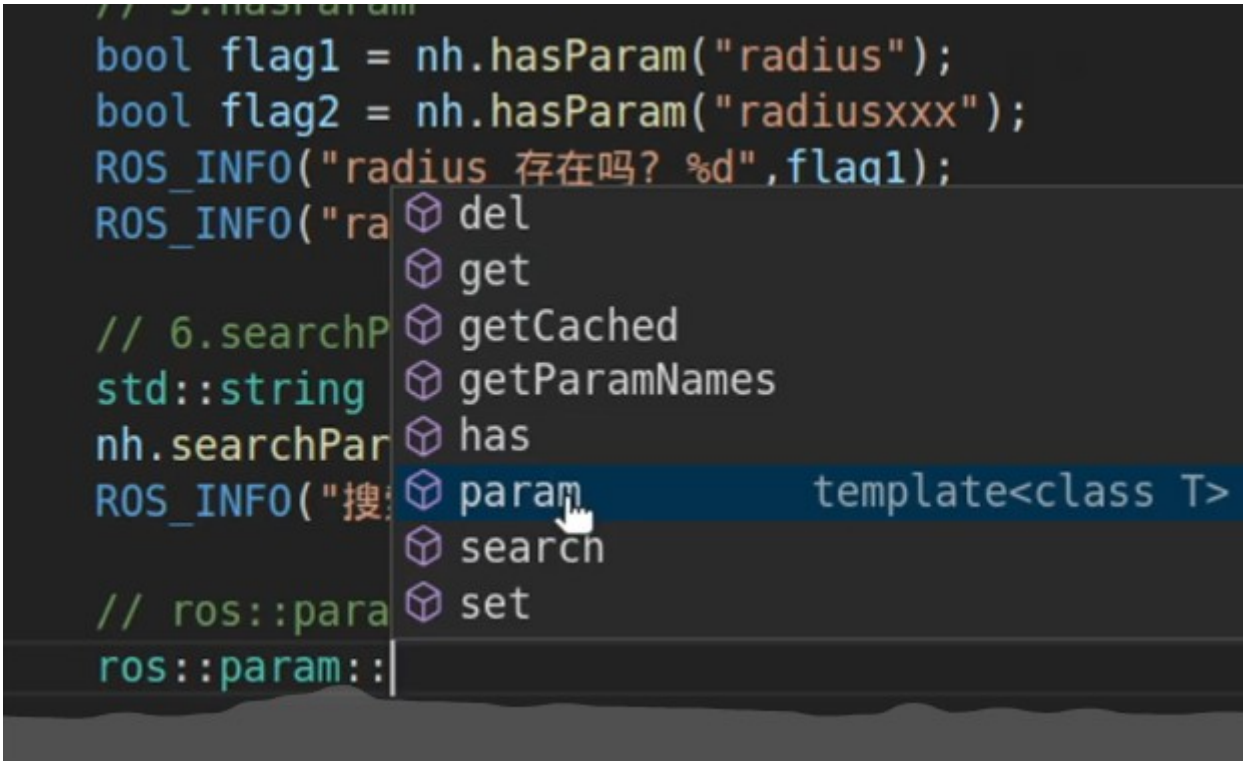
参数的查询实现 有多个实现方式

```
#include"ros/ros.h"

/*

查询实现
ros::NodeHandle

ros::param


param(键,默认值)
        存在，返回对应结果，否则返回默认值

    getParam(键,存储结果的变量)
        存在,返回 true,且将值赋值给参数2
        若果键不存在，那么返回值为 false，且不为参数2赋值

    getParamCached键,存储结果的变量)--提高变量获取效率
        存在,返回 true,且将值赋值给参数2
        若果键不存在，那么返回值为 false，且不为参数2赋值

    getParamNames(std::vector<std::string>)
        获取所有的键,并存储在参数 vector 中

    hasParam(键)
        是否包含某个键，存在返回 true，否则返回 false

    searchParam(参数1，参数2)
        搜索键，参数1是被搜索的键，参数2存储搜索结果的变量

*/
```

这个是代码的 get 的各种实现的代码
Param/ getParam/ getParamCached

```cpp
int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "");

    ros::init(argc,argv,"param_get111");
    ros::NodeHandle nh;
    double radius = nh.param("radius",0.5);
    ROS_INFO("radius = %f",radius);
    //查询键值 为radius的 没有就算0.51

    //保留几位小数

    //查询键值 为radius的 存在就算0.51
    double radius2=0.2;
    bool result = nh.getParam("radius",radius2);

    if(result)
    {
        ROS_INFO("radius1 = %f",radius2);
    }
    else
    {
        ROS_INFO("失败");
        ROS_INFO("失败  radius1 = %f",radius2);

    }
    double radius3=0.0;
    ROS_INFO("nh.getParamCached 只是性能有提升 肉眼无法观察");
    bool result1 = nh.getParamCached("radius",radius3);
    if(result1)
    {
        ROS_INFO("radius1 = %f",radius3);
    }
    else
    {
        ROS_INFO("失败");
        ROS_INFO("失败  radius1 = %f",radius3);

    }
```

param 的参数存在对应的



del 操作
nodehandle 下的 delParam

lumbing_param_server > src > demo03_param_del.cpp > main(int, char * [])

```cpp
#include "ros/ros.h"


int main(int argc, char* argv[])
{

    ros::init(argc,argv,"delparamNode");
    ros::NodeHandle nh;
    bool flag1 = nh.deleteParam("radius");
    if (flag1)
    {
        ROS_INFO("param deleted");
    }
    else{
        ROS_INFO("param not deleted");
    }
    //ros::param的实现
    bool flag2 = ros::param::del("radius_param");
    if (flag2)
    {
        ROS_INFO("param deleted radius_param");
    }
    else
    {
        ROS_INFO("param not deleted");
    }
    return 0;
}
```

setparam 后
第一次删除 删除成功
第二次 再删除 就失败了 因为在第一次已经删除了

```
问题 4  输出  调试控制台  终端  端口  注释
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ source ./devel/setup.bash
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosrun  plumbing_param_server demo03_param_del
 [ INFO] [1710658362.731564552]: param deleted
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ source ./devel/setup.bash
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosrun  plumbing_param_server demo03_param_del
 [ INFO] [1710658464.088926775]: param not deleted
 [ INFO] [1710658464.089533473]: param deleted radius_param
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosrun  plumbing_param_server demo03_param_del
 [ INFO] [1710658472.637484727]: param not deleted
 [ INFO] [1710658472.637942847]: param not deleted
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$
```