PyHEP 2023

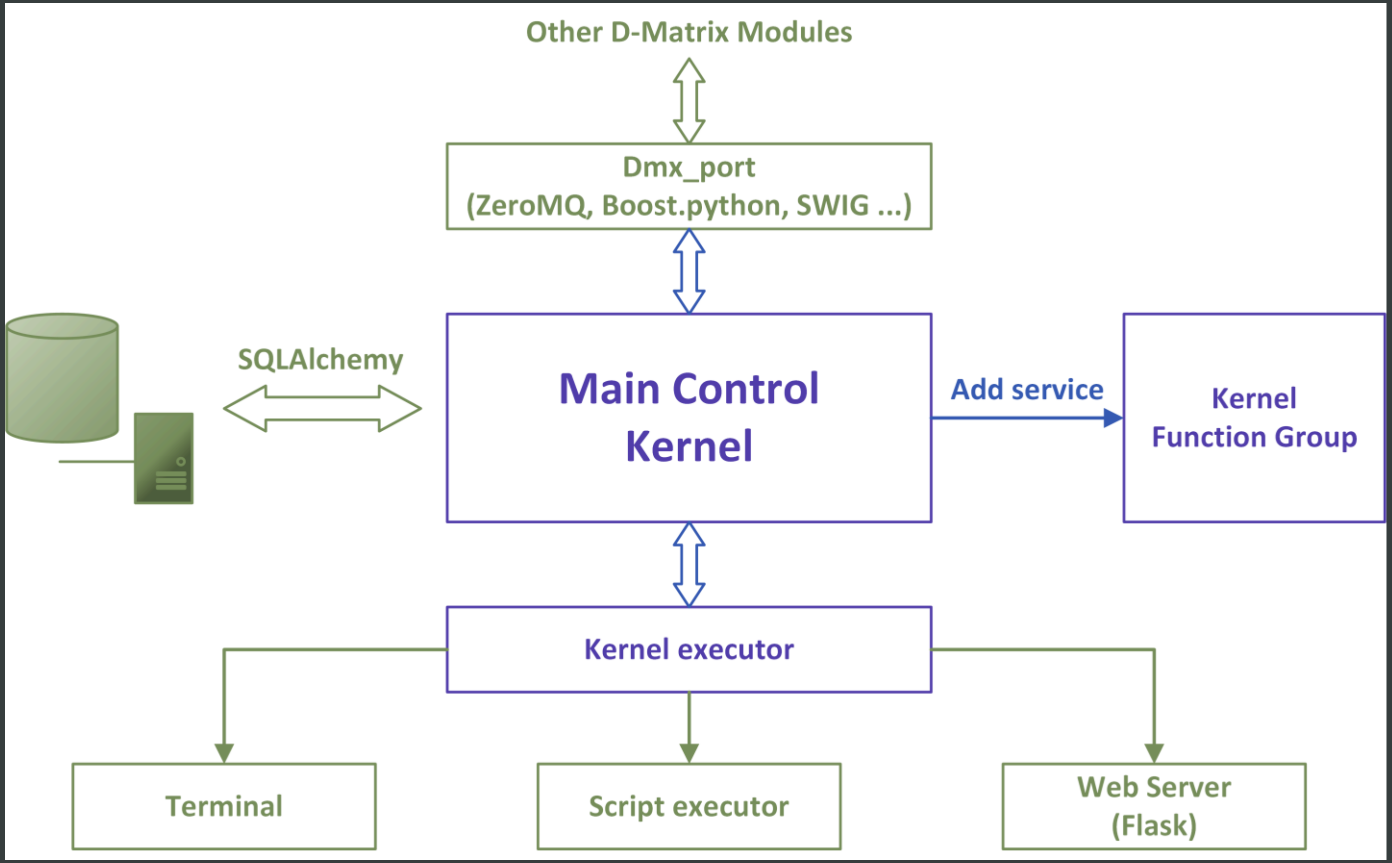
A Generic Main Control Software Structure in a Distributed Data Acquisition Platform: D-Matrix

中国科学技术大学
University of Science and Technology of China

Zhengyang Sun, Junfeng Yang
DAQLab@USTC 2023/10/09

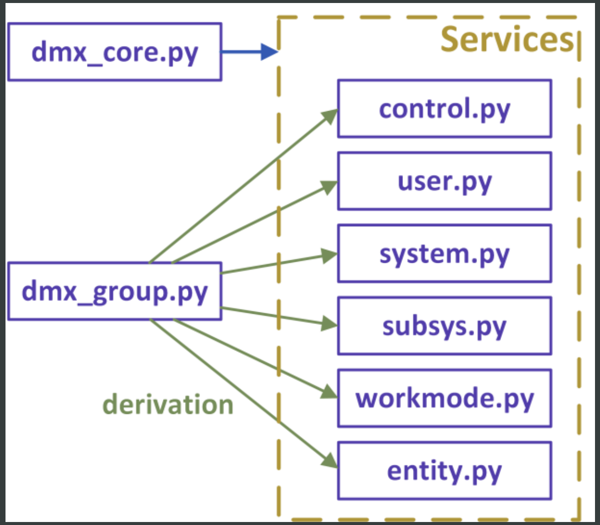
Framework and feature

- Monitoring and controlling tasks → A generic main control software
- Function Group + Executor
- Function Group: Scan Python files and add services → Facilitate the addition of function groups or functionalities
- Executor: Multiple possible executors (Terminal Command Line or Script or Web Server) → offering flexible usage options



Function Group

- Operations within the same domain exist as a group
- Each group inherits from dmx_group, and all methods within the group are considered as supported operations for that group (excluding built-in, private, and protected methods).
- Each group's Python file is located within the “services” directory. dmx_core will search all Python files in the folder and register the file names as group names.
- Enable users to easily add groups or add operations within a group, with good scalability.



Function Group

```python
class dmx_group:
    def __init__(self):
        self.group = None
        self.operations = self.__methods()

    def __methods(self):
        ops = []
        for m in dir(self):
            if m.startswith("_") or m.startswith("__"):
                continue
            if not callable(getattr(self, m)):
                continue
            ops.append(m)
        return ops

    def _execute(self, op, *args, **kwargs):
        if op not in self.operations:
            raise Exception(“%s is not a valid operation for the group %s.
Valid operations include: \n\t%s" % (op, self.group,
"\n\t".join(self.operations)))
        return self._execute_no_check(op, *args, **kwargs)

    def _execute_no_check(self, op, *args, **kwargs):
        method = getattr(self, op)
        return method(*args,  **kwargs)
```

```python
class user(dmx_group):

    def __init__(self):
        super(user, self).__init__()
        self.group = self.__class__.__name__
        self._init()

    #############################
    @authentication(["dmx_runner"])
    def get_login_users(self):
        '''
        Retrieve session information for all currently
        logged-in users.
        Command format:  user get_login_users
        Operation permissions:  dmx_runner
        '''
        pass
```

```python
class dmx_core:

    def __init__(self):
        self.groups = self.__get_groups()

    def __get_groups(self):
        groups = []
        pkgpath = os.path.dirname(__file__)+"/services"
        for _, file, _ in pkgutil.iter_modules([pkgpath]):
            groups.append(file)
        return groups
```

Executor

- Terminal Command Line:
  - Run the command: $ ./dmx_console, enter the command in the following format
  - Command format: group_name + command_name [+ args/awargs]
  - For example: user login admin, user update YYY full_name= ‘Junfeng Yang’
- Single Command Line:
  - Run the command: $ ./dmx_console group_name command_name [args/awargs]
  - For example: $ ./dmx_console user add jfyang
- Web Server (most use):
  - Run $ ./dmx_server, then a flask server will be started at localhost:5000
- Script Executor:
  - Run $ ./dmx_autorun script_dir, then automatically execute each command within the script

PyHEP 2023

Thank you

中国科学技术大学
University of Science and Technology of China

Zhengyang Sun, Junfeng Yang
DAQLab@USTC 2023/10/09