快速使用指南

百度量子计算研究所

January 27, 2021

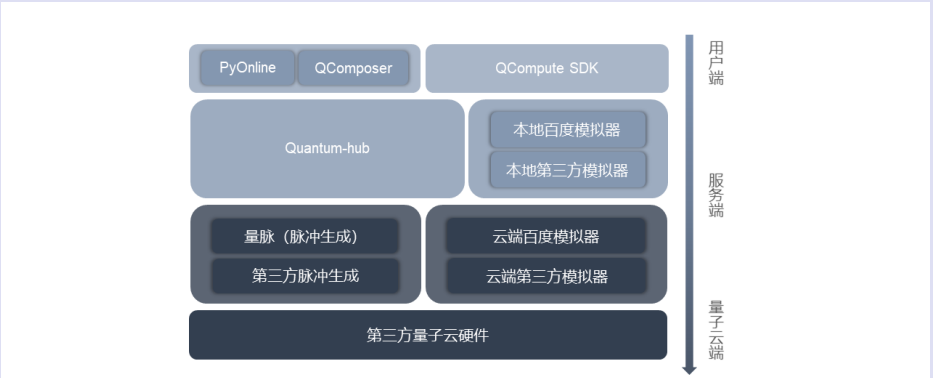
1 总览

量易伏简介

量易伏 (英文名：Quantum Leaf) 是百度研究院旗下量子计算研究所开发的云原生 (Cloud Native) 量子计算平台，提供以量子设施即服务 (Quantum infrastructure as a Service 简写为 QaaS) 模式的量子计算环境。量易伏为用户提供了三种方式进行量子计算实验，其中 QCompute 支持本地运行和连接在线资源运行，PyOnline 和 QComposer 则需要登录 Quantum-hub 在线运行。

量易伏架构

量易伏架构图



1. 用户端可以选用三类不同组件实现编程

   (a) 灵活的 QCompute
   用户需下载安装 QCompute 并进行相应环境配置才能使用。区别于另两者的是，由于携带本地模拟器，QCompute 支持量子电路的本地运行。

   (b) 方便的 PyOnline
   用户可以在 PyOnline 上使用 Python 和 QASM 语言混合编程。PyOnline 含有丰富的量子电路模板。同时，其高效可扩展架构保证用户能够上传自定义文件，下载产生的文件，定义需要使用的安装包。

   (c) 直观的 QComposer
   通过拖动量子门图标和 QASM 语言编程相结合的方式，用户可以在 QComposer 上完成可视化的量子电路搭建。

2. 量子端有本地模拟器、在线模拟器和 QPU 供选用

   (a) 用户可以使用 QCompute 自带的本地模拟器。

   (b) 使用在线模拟器需要进行 Token 验证，用户可以从 Quantum-hub 获取 Token。如果用户使用 QCompute 需自行传入 Token 参数；如果使用 PyOnline 和 QComposer 则不需要自行传入 Token 参数，因为 Quantum-hub 会自动调用。

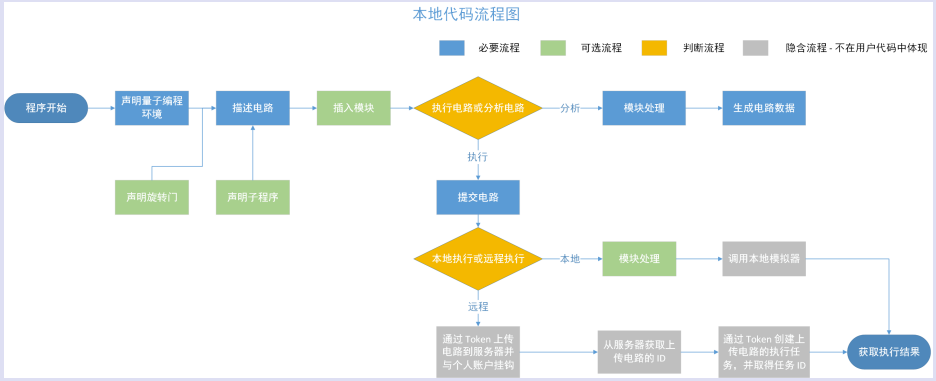
   (c) QPU 是调用第三方量子计算机的通用接口，VIP **用户有优先享有 QPU 测试使用的权利**。

   (d) 本平台可用的量子端如下表所示：

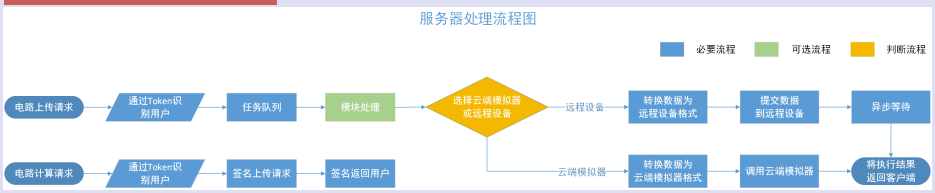
类型	量子端	说明
本地	LocalBaiduSim2	使用 Python 编写的 Sim2 本地版
云端	CloudBaiduSim2	百度模拟器云版（曾用的后端名）
云端	CloudBaiduSim2Water	使用 Python 编写的多实例 Sim2 模拟器云版
云端	CloudBaiduSim2Earth	使用 Python 编写的单一实例高配置 Sim2 模拟器云版
云端	CloudBaiduSim2Thunder	使用 C++ 编写的单一实例高配置 Sim2 模拟器云版
云端	CloudBaiduSim2Heaven	使用 C++ 编写的单一实例集群 Sim2 模拟器云版
云端	CloudAerAtBD	开源 Aer(C++ 版) 模拟器云版
云端	CloudQpu	VIP 用户通过 QPU 方式使用第三方量子计算机

相关流程

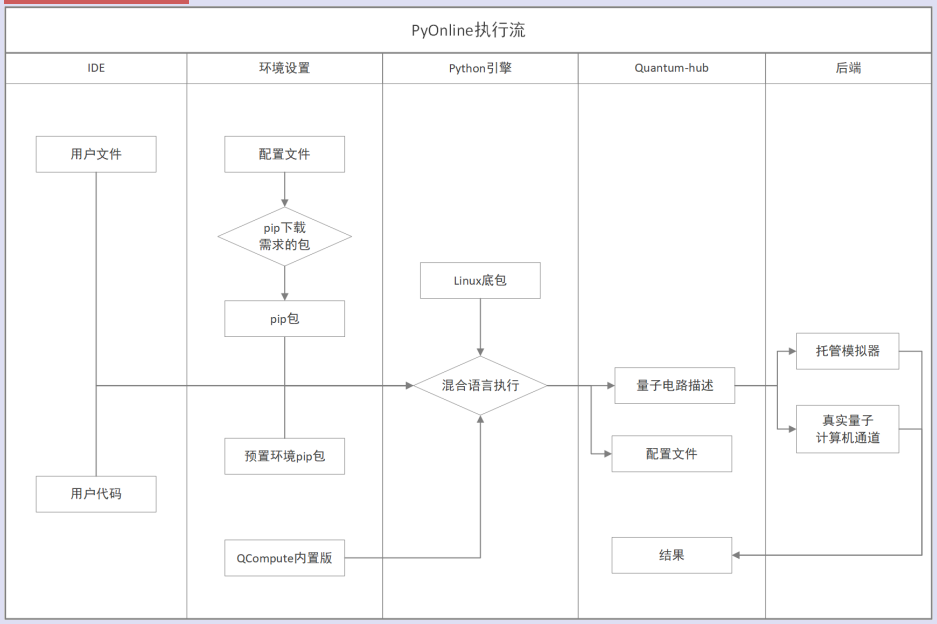
QCompute 本地代码流程图



QCompute 服务器处理流程图



PyOnline 工作流图



2 代码示例

前言

QCompute 使用方法参照 QCompute，相应示例代码和用户手册随安装包下载供用户使用。
如用户对 PyOnline 和 QComposer 的基本操作和功能情况有疑惑，请查看相应页面的 **【帮助说明】**，还可以通过 **【反馈建议】**或 **量易伏用户 QQ 群**（1147781135）寻求帮助。

PyOnline 示例

```
from QCompute import *

# 新建一个量子环境
env = QuantumEnvironment()
# 量子端可自选，本例使用百度自研的量子后端进行模拟
env.backend(BackendName.CloudBaiduSim2)

# 初始化6个量子比特，其处于基态|000000>
q = [env.Q[i] for i in range(6)]
# 量子比特下标从0开始，在q[2]上添加 Hadamard 门
H(q[2])
# 添加 CNOT 门，其中q[2]为控制量子比特，q[4]为被控量子比特
CX(q[2], q[4])

# 测量函数 MeasureZ() 支持用户进行个别比特的测量
# 本例测量 q[2], q[4], q[5], range(3) 代表测量3个量子比特
MeasureZ([q[2], q[4], q[5]], range(3))
# 设定shot值为2048，表示测量该电路2048次
# 设定downloadResult为False，表示结果不返回
# 请在 Quantum-hub 的【查看电路】和【查看任务】中查看
taskResult = env.commit(2048, downloadResult=False)
```

输出结果：“011”:1038 “000”:1010
注意：量易伏采用大端在前的标记方法进行输出。因此，本示例结果的输出顺序为 q[5] q[4] q[2]。

QComposer 示例

注意：QComposer 行注释使用 //。

```
OPENQASM 2.0;
include "qelib1.inc";
// 初始化6个量子比特和相应的6个经典比特
qreg q[6];
creg c[6];
// 添加 Hadamard 门，作用于q[0]
h q[0]
// 添加 CNOT 门，q[0] 为控制量子比特，q[1] 为受控量子比特
cx q[0], q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

输出结果："11":515 "00":509

QComposer 实现了量子编程的可视化，以上代码仅需拖动量子门图标即可构建，不需要用户编写 QASM 代码。接下来展示的子程序功能将帮助用户对自己即将构建的量子电路形成全局的认识并需要用户进行少量的代码编写。**子程序** (Subroutines) 功能在 QComposer 中扮演着函数的角色，它使得量子编程中的参数和量子寄存器使用更灵活。子程序的使用将使用户的量子编程更加得心应手，示例代码如下。

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[6];
creg c[6];
// 声明子程序nG0，它有3个参数param0、param1、param2
// 并且其操作的量子寄存器由qb0、qb1、qb2指定
gate nG0(param0, param1, param2) qb0,qb1,qb2
{
    // rzz门使用参数param0、param1、param2
    // 并作用在qb0、qb1所指定的量子位上
    rzz(param0, param1, param2) qb0, qb1;
}
gate nG1(param0, param1, param2) qb0,qb1,qb2
{
    // 调用子程序nG0，并通过四则运算来灵活地使用参数
    nG0(param0/2, param1/4, sin(param2))  qb0, qb1, qb2;
}
// 调用子程序nG1，param0 = param1 = param2 = pi
// 指定其操作作用于q[0]、q[1]、q[2]
nG1(pi, pi, pi)  q[0], q[1], q[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

输出结果："10":506 "00":518

QComposer 示例

在这里我们明确子程序的使用规则和注意事项。

1. 子程序使用规则
   (a) 子程序分为声明和调用两个部分，子程序声明必须在子程序调用的前面。
   (b) 子程序声明时，代码块以 gate 关键字开头，后接子程序名和圆括号 () 以及量子寄存器，通过拖拽量子门图标或者编写 QASM 代码构建量子电路。
   (c) 任何传入参数都应在圆括号中声明，这些参数名称是传入参数的占位符。当圆括号中参数名称为空时，QComposer 将自动补全为 gate name(param0) q，但这不影响子程序的使用，即在声明子程序的参数名称和调用子程序的传入参数可以同时为空；在参数名称非空时，调用子程序时传入参数数量应与子程序参数数量保持一致。
   (d) qb0,qb1,qb2 是量子寄存器的占位符。当调用子程序时，必须指明被操作的量子寄存器。例如，nG1(pi, pi, pi) q[0], q[1], q[2] 中指明了将 nG1 作用在 q[0],q[1],q[2] 这 3 个量子寄存器上。

2. 子程序注意事项

```
gate nG1(param0, param1, param2) qb0,qb1,qb2
{
    nG0(param0/2, param1/4, sin(param2))  qb0, qb1, qb2;
}
```

在 nG1 的子程序声明代码中，子程序 nG1 调用了 nG0。在这个代码块中试图更改 nG0 的量子门是不合法的。可以对 nG0 的参数进行四则运算或者设置默认值，但不能对 nG1 的参数进行四则运算或者设置默认值。由此总结得到：

   (a) **子程序中不能进行测量**。
   (b) 仅在子程序声明部分可以修改本子程序的量子门，在调用阶段无法对被调用子程序内的量子门进行修改。
   (c) 仅在子程序调用部分可以对被调用子程序的参数部分进行四则运算或者设置默认值，在声明阶段无法对子程序参数进行四则运算或者设置默认值。
   (d) 子程序在使用参数和量子寄存器时的灵活性体现在于

```
gate nG1(param0, param1, param2) qb0,qb1,qb2
{
    // 参数与量子寄存器的位置可变
    nG0(param2/2, param0/4, sin(param1))  qb1, qb0, qb2;
}
```

FAQ

1. 如何在量易伏上编程？
   (a) 用户可以混合使用 Python 编程语言和 QASM 进行实验和开发；
   (b) 如用户使用 Quantum-hub 上的 QComposer 或 PyOnline 进行实验，平台自动调用 Token，不需要用户自行填入；
   (c) 如用户使用 QCompute 进行实验，需要传入 Token 以连接远端模拟器或设备。
2. 为什么要用 QCompute ？
   (a) 可以迁移使用经典语言知识；
   (b) 非联网的情况下依然可以使用本地模拟器。
3. 本地模拟器和在线模拟器有什么区别？
   (a) 本地模拟器使用本地资源，在线模拟器使用百度云计算资源；
   (b) 在线模拟器在百度云计算的支持下提供更为强大的算力。
4. 如何加速计算过程？
   (a) 使用在线模拟器；
   (b) 如果是交互式计算，考虑使用并行计算方法。
5. Credit 的用处？
   (a) 在线资源计算的时候每次会消耗 1 点，PyOnline 本身的运行将额外扣除 1 点；
   (b) 获取更多点数请通过邮箱 quantum@baidu.com 或者 Quantum-hub 中的 **【反馈建议】**联系我们。
6. 如何注册 VIP？

   VIP 注册为邀请制，用户需要完成：
   (a) 相应身份核验；
   (b) 提交 VIP 申请到邮箱 quantum@baidu.com 或 Quantum-hub 中的 **【反馈建议】**;
   (c) 待后台判断通过，则申请成功。
7. Quantum-hub 与 QCompute 的区别和联系？
   (a) Quantum-hub 是量易伏中控端，无论云端 PyOnline , QComposer 或是本地运行的 QCompute 结果都可以从 Quantum-hub 获取；
   (b) QCompute 是量易伏的一个高级开发组件，通过此组件，用户可以进行 Python 与简单的量子编程语言的混合编程等。