Guía paso a paso: Ejecución de un circuito cuántico en Intel Quantum SDK usando Docker y VS Code

Sadi Nicolás Mendoza Soriano

Proyecto: *Quantum Machine Learning Using Electronic Spins*

Introducción

Esta guía está pensada para que puedas aprender de forma práctica y comprensible cómo ejecutar un circuito cuántico básico utilizando el **Intel Quantum SDK** dentro de un contenedor **Docker**, trabajando directamente desde **Visual Studio Code**.

El propósito es que entiendas el flujo completo del desarrollo cuántico: desde la preparación del entorno hasta la ejecución del código y la interpretación de resultados, sin depender de editores de texto de consola.

1. Configuración inicial del entorno

1. Abrir el contenedor Docker

Primero, abre una terminal (CMD, PowerShell o VS Code Terminal) y ejecuta:

```
docker run -it --name qml_intel \
  -v "D:\usuario\Desktop\qml_spines_docker_intel_sdk:/workspace" \
  intellabs/intel_quantum_sdk:latest /bin/bash
```

Esto inicia un contenedor con el Intel Quantum SDK y monta tu carpeta local del proyecto en la ruta ‘/workspace‘.

2. Conectar VS Code al contenedor

Dentro de Visual Studio Code:

1. Instala la extensión **Remote - Containers** (de Microsoft).
2. Abre la carpeta local del proyecto:

```
D:\usuario\Desktop\qml_spines_docker_intel_sdk
```

3. Presiona `Ctrl + Shift + P` y selecciona **Remote-Containers: Attach to Running Container**.

4. Elige el contenedor llamado.

VS Code ahora trabajará directamente dentro del entorno de Docker, con acceso a todas las herramientas del Intel Quantum SDK.

2. Creación del programa cuántico

Con el entorno listo:

1. En VS Code, crea una nueva carpeta llamada 'ejemplo$_c$*untico*'.

1. Dentro de ella, crea un archivo llamado:

```
ap_1.cpp
```

2. Copia el siguiente código y guárdalo.

Listing 1: Programa cuántico básico en Intel Quantum SDK

```
#include <iostream>                                    // Entrada/
   salida est ndar
#include <vector>                                      // Vectores
   din micos
#include <bitset>                                      // Mostrar
   estados binarios
#include <clang/Quantum/quintrinsics.h>                // Librer a de
    compuertas cu nticas
#include <quantum_full_state_simulator_backend.h>      // Backend del
   simulador cu ntico

const int N = 3;
qbit q[N];        // C bits del circuito
cbit c[N];        // Bits cl sicos para almacenar mediciones
double param[2];  // Par metros para las rotaciones

// Kernel cu ntico: inicializa los c bits en |0>
```

```
13 quantum_kernel void prepare_all() {
14     for (int i = 0; i < N; i++) {
15         PrepZ(q[i]);
16     }
17 }
18
19 // Kernel cu ntico: mide todos los c bits
20 quantum_kernel void measure_all() {
21     for (int i = 0; i < N; i++) {
22         MeasZ(q[i], c[i]);
23     }
24 }
25
26 // Kernel con rotaciones personalizadas
27 quantum_kernel void custom_operation(qbit &q1, qbit &q2) {
28     RX(q1, param[0]);
29     RX(q2, param[1]);
30 }
31
32 // Kernel principal
33 quantum_kernel void test_function() {
34     custom_operation(q[0], q[1]);
35 }
36
37 int main() {
38     iqsdk::IqsConfig iqs_config(N, "noiseless");
39     iqsdk::FullStateSimulator iqs_device(iqs_config);
40
41     if (iqsdk::QRT_ERROR_SUCCESS != iqs_device.ready()) {
42         std::cerr << "Error: el simulador no se pudo inicializar."
              << std::endl;
43         return 1;
44     }
45
46     prepare_all();
47     param[0] = 15.0;
48     param[1] = 30.0;
49     test_function();
50     measure_all();
51
52     std::cout << "\n--- Resultados de medici n ---" << std::endl;
```

```
    for (int i = 0; i < N; i++) {
        std::cout << "Qubit " << i << " -> " << (bool)c[i] << std::endl;
    }

    std::cout << "\n--- Probabilidades de cada estado ---" << std::endl;
    std::vector<std::reference_wrapper<qbit>> qids;
    for (int i = 0; i < N; i++) qids.push_back(std::ref(q[i]));

    std::vector<iqsdk::QssIndex> bases;
    for (int i = 0; i < (1 << N); i++) {
        std::string bits = std::bitset<8>(i).to_string();
        bits = bits.substr(bits.size() - N);
        iqsdk::QssIndex idx(bits);
        bases.push_back(idx);
    }

    iqsdk::QssMap<double> prob_map = iqs_device.getProbabilities(qids, bases);
    for (auto &entry : prob_map) {
        std::cout << "|" << entry.first << "> : " << entry.second << std::endl;
    }

    std::cout << "\nEjecuci n finalizada correctamente." << std::endl;
    return 0;
}
```

3. Compilar y ejecutar desde VS Code

Abre una terminal integrada en VS Code (menú `Terminal >New Terminal`) y ejecuta los siguientes comandos dentro del contenedor:

1. Compilación del programa

```
../intel-quantum-compiler ap_1.cpp -o ap_1
```

2. Ejecución del circuito

```
./ap_1
```

Si todo está correcto, verás algo similar a:

```
--- Resultados de medición ---
Qubit 0 -> 1
Qubit 1 -> 0
Qubit 2 -> 1

--- Probabilidades de cada estado ---
|000> : 0.12
|001> : 0.08
|010> : 0.24
|011> : 0.56
```