順序付けの問題

この記事は、インテル® デベロッパー・ゾーンに公開されている「Ordering issues」の日本語参考訳です。

この記事は、インテル® スレッディング・ビルディング・ブロック (インテル® TBB) 4.4 Update 2 以降で利用可能な新しいノード opencl_node について説明するシリーズのパート 5 です。このノードは、インテル® TBB のフローグラフで OpenCL* デバイスの利用と連携を可能にします。このシリーズのパート 1 はこちらからご覧いただけます。

パート 4 では、OpenCL* プログラムを指定し、プログラムからカーネルを選択して、カーネル呼び出しに引数をバインドする方法を説明しました。この記事では、順序付けの問題を回避するため、型指定したメッセージキーの使用法について述べます。

順序付けの問題

次のサンプルコードは、バッファーを作成し、同じ値を格納する 2 つの function_node を作成します。opencl_node は、バッファーのペアを受け取り、乗算を行います。

```
#define TBB_PREVIEW_FLOW_GRAPH_NODES 1
#include "tbb/flow_graph_opencl_node.h"

#include <cmath>
#include <stdexcept>
#include <string>

int main() {
    try {
        using namespace tbb::flow;
        typedef opencl_buffer<cl_int> CLBuf;
        const int N = 10;

        opencl_graph g;
        opencl_node<tuple<CLBuf,CLBuf>> clMul( g, "mul.cl", "mul" );
        clMul.set_ndranges( { N } );

        function_node<int,CLBuf> filler0( g, unlimited, [&g,N]( int i ) -> CLBuf
{
            CLBuf b(g, N);
            std::fill( b.begin(), b.end(), i );
            return b;
        } );

        function_node<int,CLBuf> filler1 = filler0;

        function_node<CLBuf> checker( g, serial, []( const CLBuf &b ) {
            for ( cl_int v : b ) {
                int r = int(std::sqrt(v) + .5);
                if ( r*r != v )
                    throw std::runtime_error( std::to_string(v) + " is not a
square of any integer number" );
            }
        } );
```

```
        make_edge( filler0, input_port<0>(clMul) );
        make_edge( filler1, input_port<1>(clMul) );
        make_edge( output_port<0>(clMul), checker );

        for ( int i = 0; i<1000; ++i ) {
            filler0.try_put( i );
            filler1.try_put( i );
        }
        g.wait_for_all();
    } catch ( std::exception &e ) {
        std::cerr << "An exception has occurred: " << e.what() << std::endl;
    }
    return 0;
}
mul.cl:
kernel void mul( global int* b1, global int* b2 ) {
    const int index = get_global_id(0);
    b1[index] *= b2 [index];
}
```

チェッカーノードは、opencl_node が値を含むバッファーを受け取り、乗算を行って値の 2 乗を生成することを想定します。しかし、このサンプルは失敗する可能性があります。

```
An exception has occurred: 54522 is not a square of any integer number
```

1 番の問題は、function_node が同時に実行し、非決定的な順序でメッセージを送信するため、opencl_node が異なる値を含むバッファーを乗算してしまう可能性があります。この問題に対応するため、opencl_node はインテル® TBB 4.4 Update 2 で追加された型指定によるキーのマッチング機能をサポートしています。

この機能を有効にするには、opencl_node 作成時にテンプレート引数として key_matching<Key> ポリシーを指定します。

```
opencl_node<tuple<CLBuf,CLBuf>, key_matching<int>> clMul(g, "mul.cl", "mul");
```

さらに、メッセージ型が "型指定したメッセージキー" のコンセプトの要件を満たしていなければなりません (詳細は、インテル® TBB ドキュメントの「join_node の型指定したメッセージキー」を参照してください)。そのため、opencl_buffer<cl_int> クラスを "int key() const" メソッドで拡張しました。

```
class CLBuf : public opencl_buffer<cl_int> {
    int my_key;
public:
    CLBuf() {}
    CLBuf( opencl_graph &g, size_t N, int k )
        : opencl_buffer<cl_int>(g, N), my_key(k) {}
    int key() const { return my_key; }
};
```

このメソッドは、受け取ったメッセージを正しくマッチングするため、opencl_node によって呼び出されます。ノードは、すべての入力ポートに渡すため、同じキー値のメッセージを待機します。

サンプルコードのほかの部分に変更はありません。

```
#define TBB_PREVIEW_FLOW_GRAPH_NODES 1
#include "tbb/flow_graph_opencl_node.h"

#include <cmath>
#include <stdexcept>
#include <string>

using namespace tbb::flow;

class CLBuf : public opencl_buffer<cl_int> {
    int my_key;
public:
    CLBuf() {}
    CLBuf( opencl_graph &g, size_t N, int k ) : opencl_buffer<cl_int>(g, N),
my_key(k) {}
    int key() const { return my_key; }
};

int main() {
    try {
        using namespace tbb::flow;
        const int N = 10;

        opencl_graph g;
        opencl_node<tuple<CLBuf,CLBuf>, key_matching<int>> clMul( g, "mul.cl",
"mul" );
        clMul.set_ndranges( { N } );

        function_node<int,CLBuf> filler0( g, unlimited, [&g,N]( int i ) -> CLBuf
{
            CLBuf b(g, N, i); // 最後の引数がキー値
            std::fill( b.begin(), b.end(), i );
            return b;
        } );

        function_node<int,CLBuf> filler1 = filler0;

        function_node<CLBuf> checker( g, serial, []( const CLBuf &b ) {
            for ( cl_int v : b ) {
                int r = int(std::sqrt(v) + .5);
                if ( r*r != v )
                    throw std::runtime_error( std::to_string(v) + " is not a
square of any integer number" );
            }
        } );

        make_edge( filler0, input_port<0>(clMul) );
        make_edge( filler1, input_port<1>(clMul) );
        make_edge( output_port<0>(clMul), checker );

        for ( int i = 0; i<1000; ++i ) {
            filler0.try_put( i );
            filler1.try_put( i );
        }
        g.wait_for_all();
    } catch ( std::exception &e ) {
        std::cerr << "An exception has occurred: " << e.what() << std::endl;
    }
    return 0;
}
```

これで、サンプルコードが想定どおりに動作するようになりました。

次の点にも留意すべきです。

- opencl_node に対してキーのマッチングポリシーが指定されている場合、そのすべてのメッセージ型が "型指定したメッセージキー" をサポートしなければなりません。
- 型が構造体で、opencl_buffer から継承されない場合、OpenCL* にはそのまま渡されます。そのため、カーネルはその関数パラメーターを適切に宣言しなければなりません。

これで、この opencl_node シリーズは終わりです。このシリーズのパート 1 はこちらからご覧いただけます。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。