connection.hpp

这段代码定义了一个名为 `rdmalib` 的命名空间，其中包含名为 `Connection` 的结构体和一些与 RDMA 通信相关的类型和枚举。该结构体对 RDMA 连接进行了封装，并使用了不同的 RDMA 库（libfabric 或 Infiniband Verbs）。 在 `Connection` 结构体中，定义了一些成员变量和成员函数，其中包括连接的状态、连接的生存期、用于 RDMA 操作的 Queue Pair、用于通知完成操作的 Completion Queue 等等。此外，还定义了连接的初始化和关闭方法，以及发送、接收和写入数据的方法等等。最后，该代码包含了一些头文件和宏的定义。

connection.cpp

这是一个名为“rdmalib”的库的 C++ 代码，它提供了一个高级接口，用于使用远程直接内存访问 (RDMA) 网络。 RDMA 是一种允许应用程序直接访问远程内存而无需涉及 CPU 的技术。 该库包括用于管理 RDMA 连接、发布发送和接收以及轮询工作完成的类。 该库还包括用于发布写入、比较和交换操作以及获取和添加操作的函数。 该代码使用了多个外部库，包括 libfabric，它是一个用户级网络堆栈，旨在以统一的方式提供对多个网络结构的访问。 该代码还使用 Spdlog 库进行日志记录，使用 fmt 库格式化日志消息。

这是一个C++程序文件，文件名为connection.cpp，属于rdmalib库的一部分，提供了一些方便使用RDMA的接口。程序包含了Connection和ConnectionConfiguration两个类，Connection类是连接抽象的表示，ConnectionConfiguration类是连接的一些配置项。程序实现了RDMA连接建立、连接关闭、发送和接收数据等功能。其中用到了IBVERBS（Infiniband Verbs）库提供的一些接口。

rdmalib.hpp

该代码是 C++ 语言程序，定义了一个命名空间`rdmalib`，其中包含了 RDMA（Remote Direct Memory Access）通信库的相关实现。 代码中定义了一些结构体，如`Configuration`、`Address`、`RDMAActive`和`RDMAPassive`等。其中，`Configuration`中包含了一些配置信息，如*crn_info_handle_t*、cookie和_credential等。`Address`中包含了 IP 地址和端口等信息。`RDMAActive`和`RDMAPassive`分别表示主动端和被动端，用于建立 RDMA 连接。 在这个命名空间中，还定义了一些函数和变量，并通过条件编译（`#ifdef`）来区分不同操作系统的实现。其中`USE_LIBFABRIC`表示是否使用 Fabric 库。`USE_GNI_AUTH`表示是否使用 GNI 认证。 整段代码通过头文件保护机制（`#ifndef`和`#define`）来避免重复包含。

rdmalib.cpp

该文件是rFaaS平台中的一个组件rdmalib的源代码文件，提供了RDMA连接所需的工具和配置类。该文件主要包含了如下内容：

- 连接配置类Configuration：定义了用于配置RDMA连接所需的一些参数，比如cookie等；
- 目标机地址类Address：定义了连接对端地址，包括IP地址和端口号；
- 主动端RDMAActive：定义了主动端连接所需的一些方法和参数，比如连接和断开连接等；
- 被动端RDMAPassive：定义了被动端连接所需的一些方法和参数，包括监听和关闭监听等。

该文件主要是用于实现RDMA连接的配置和创建，为rFaaS平台的RDMA连接提供支持。

recv_buffer.hpp

这是一个 C++ 的头文件，其中定义了名为 `rdmalib::RecvBuffer` 的结构体。此结构体代表了在 RDMA 库中用于接收数据的缓冲区。头文件中包含以下成员变量：

- `_rcv_buf_size`：一个整数，表示接收缓冲区的大小。 -
- `_refill_threshold`：一个整数，表示当未完成的接收请求数量低于此阈值时需要重新填充缓冲区。 - `_requests`：一个整数，表示未完成的接收请求数量。 -
- `_conn`：指向表示与接收缓冲区相关联的连接的 `rdmalib::Connection` 对象的指针。 此结构体包含以下成员函数： -
- `RecvBuffer(int rcv_buf_size)`：构造函数，使用给定的 `rcv_buf_size` 初始化 `_rcv_buf_size`、`std::min(_rcv_buf_size, DEFAULT_REFILL_THRESHOLD)` 和 0 分别为 `_refill_threshold` 和 `_requests`，同时将 `_conn` 初始化为 null。 -
- `void connect(rdmalib::Connection * conn)`：将 `_conn` 设置为给定的 `conn`，将 `_requests` 重置为 0，并调用 `refill()` 函数。 -
- `poll(bool blocking = false)`：轮询与接收队列相关联的完成队列，并返回一个包含完成数据的指针和完成数的元组。如果 `blocking` 设置为 true，则该函数会阻止，直到至少一个完成可用。该函数还会将 `_requests` 减去完成数。 -
- `refill()`：如果未完成的接收请求数量低于重新填充阈值，则重新填充接收缓冲区。该函数向与连接相关联的队列中发布了一批空接收请求，并将 `_requests` 成员变量更新为 `_rcv_buf_size`。

头文件还包括了 `optional` 头文件、`connection.hpp` 头文件和来自 SPDLog 日志库的 `spdlog.h` 头文件。头文件通过包含 `ifndef` 指令的头文件保护指令防止多次包含。

util.hpp

util.cpp

该文件是rFaas.tar.gz.extract/rFaaS/rdmalib/lib目录下的util.cpp文件，包含一些函数的实现，具体如下：

- expect_true(bool flag): 断言flag为true，若不是，则会抛出一个assertion_fail异常。
- expect_false(bool flag): 断言flag为false，若不是，则会抛出一个assertion_fail异常。
- traceback(): 打印函数调用的堆栈信息。该函数利用execinfo.h库提供的backtrace和backtrace_symbols函数获取堆栈信息，并利用spdlog库打印信息。

该文件是rFaas.tar.gz.extract/rFaaS/rdmalib/lib目录下的util.cpp文件，包含一些函数的实现，具体如下：

- expect_true(bool flag): 断言flag为true，若不是，则会抛出一个assertion_fail异常。
- expect_false(bool flag): 断言flag为false，若不是，则会抛出一个assertion_fail异常。
- traceback(): 打印函数调用的堆栈信息。该函数利用execinfo.h库提供的backtrace和backtrace_symbols函数获取堆栈信息，并利用spdlog库打印信息。

buffer.cpp

本文件是rdmalib库中的buffer.cpp文件，包含了Buffer和RemoteBuffer两个类的实现。Buffer类包含了内存缓冲区的各种信息和操作，例如内存大小、内存地址、关键字等等；RemoteBuffer类则包含了一个远程内存缓冲区的信息，即远程地址和关键字。其中，Buffer类的数据成员中，*size表示缓存块的数据大小*，header表示缓存块的头大小（默认为0），*bytes表示缓存块的总大小*，byte_size表示每个数据缓存块的大小，*ptr表示实际内存块的首地址*，mr表示内存块的信息，_own_memory表示内存是否被该类实例对象所占用，是一个 bool 类型。代码实现包含了构造函数、拷贝构造函数、赋值重载函数、析构函数等等成员函数，例如Buffer::register_memory()函数用于将内存块注册准备使用 RDMA 网络传输，Buffer::sge()函数用于得到RDMA网络传输的sge信息，RemoteBuffer类则有默认构造函数和可以设置成员的构造函数。

server.hpp

该文件为rFaaS的rdmalib库中的server.cpp文件，代码实现了rdmalib中的服务器状态类ServerStatus的序列化和反序列化方法。其中使用了cereal库中的JSONInputArchive和JSONOutputArchive类进行序列化和反序列化。

client.cpp

该程序文件是rFaaS的执行管理器中用于客户端连接的C++代码文件。它包括一个名为Client的类和几个类成员函数。在该类的构造函数中，通过传入的连接对象和物理域对象，初始化客户端对象并设置内存访问权限，接收请求缓冲区等。该类还包括其他一些辅助函数，如重载队列、禁用客户端等。程序使用了rdmalib库，同时具有可移植性。

lib/executor.cpp

该程序文件是一个C++源文件，文件路径为rFaas.tar.gz.extract/rFaaS/rfaas/lib/executor.cpp。

该文件包含了rFaaS框架中executor的实现。executor是rFaaS框架中用于执行函数的组件，通过与manager和executor之间建立的RDMA连接协作完成函数的异步执行任务。

该文件包含了以下主要内容：

- 头文件引用，包括rdmalib/rdmalib.hpp、chrono、spdlog/spdlog.h、rdmalib/allocation.hpp、rdmalib/connection.hpp、rdmalib/buffer.hpp、rdmalib/util.hpp、rfaas/connection.hpp、rfaas/executor.hpp和rfaas/resources.hpp。
- polling_type和executor_state结构体，这两个结构体定义了executor内部用来协助函数执行和结果处理的一些变量和方法。
- executor类，包含了executor的主要实现。其中包括构造函数、析构函数、load_library方法、allocate方法、deallocate方法、poll_queue方法等。这些方法主要包含了executor的初始化、函数的载入、executor的连接建立、函数执行的异步处理等内容。

cli.cpp

该程序是rFaaS（Remote Function as a Service）框架的executor管理器。它是一个命令行工具，可以通过给出的参数来配置其行为。在运行时，它将读取设备数据库和管理员的配置文件，然后启动rFaaS executor管理器。当程序收到SIGINT信号时，将通过调用信号处理程序来关闭executor管理器。

executor/server.cpp

该程序文件是rFaaS系统中executor server（执行器服务器）的源代码。其包含了SignalHandler类和Server类的定义与实现。其中SignalHandler类是用于处理SIGINT信号的，而Server类中包含了该服务器的状态、连接、快速执行器、连接指针、工作完成缓冲区等重要信息，并且封装了一些用于运行该服务器的方法，例如：监听（listen()）、重新加载队列（reload_queue()）、轮询通信（poll_communication()）、轮询服务器通知（poll_server_notify()）和轮询线程（poll_threads()）等方法。其中轮询通信（poll_communication()）方法和轮询服务器通知（poll_server_notify()）方法是该服务器的核心功能。整个程序是通过RDMA进行远程数据访问来实现的。

executor/fast_executor.cpp

该程序文件是rFaas函数计算框架中的一个快速执行器，文件名为fast_executor.cpp。程序主要通过RDMA传输机制来实现远程函数调用。该文件包含了一些工具类，如SignalHandler、Thread、FastExecutors等，并且还包括了一些库文件，如rdmalib、spdlog等。程序通过不同的状态来实现轮询机制，处理来自客户端的请求，并在请求结束后将结果返回给客户端。在处理过程中，程序会记录执行时间和轮询时间，并不断地向远程管理器提交信息。

cold_benchmark.cpp

该程序文件为一个用于对 rFaaS 应用进行冷启动性能测试的基准测试程序。代码主要包括以下几个部分：

1. 引入头文件和库文件
2. 解析命令行参数并设定日志等级
3. 从文件中读取设备信息和基准测试配置信息
4. 从连接池中读取资源管理器的连接信息并创建执行器
5. 创建输入缓存和输出缓存
6. 使用 rdmalib 库中的 Benchmarker 计算基准测试指标
7. 基于执行器分配资源，并调用执行器执行函数，计算 latency 和 throughput
8. 输出基准测试结果和所有输出缓存中的结果

程序主要使用了 rdmalib 库中的函数进行远程 DMA 操作，同时也使用了 rfaas 库中的类进行 rFaaS 应用的操作。

warm_benchmark.cpp

该程序文件是一个用于测试serverless-rdma的warm_benchmark程序。程序主要功能包括读取设备信息和连接详情，配置cookie，读取基准测试设置，读取执行器的连接细节，运行warm-up，开始实际测试, 输出测试结果。程序中使用了许多第三方库，如spdlog、cxxopts等。同时，程序中使用了许多RDMA相关的库来进行双方通信。

cpp_interface.cpp

该程序文件实现了一个基于 RDMA 技术的无服务器计算 C++ 接口，主要包括以下功能：

1. 读取设备信息和基准配置信息，以及连接在远程执行器上的执行器的连接细节，以便为无服务器计算运行进行准备；
2. 初始化输入和输出缓冲区，执行阻塞和非阻塞计算，以及混合阻塞和非阻塞计算；
3. 输出计算结果，包括执行成功与否，输出缓冲区中每个位置上存储的数据等。

其主要核心是通过连接在远程执行器上的空闲资源，将传统计算应用转化为无服务器计算应用来，在避免了对物理机进行大规模繁琐的搭建和维护的同时，提高了计算效率。

scalable_benchmark.cpp

该程序文件为C++代码文件，文件名为scalable_benchmark.cpp，该文件属于rFaas.tar.gz.extract/rFaaS/benchmarks目录下。该程序主要实现了一个基于MPI和RDMA的可扩展基准测试。程序首先从命令行参数中读取配置信息，并通过RDMA和 MPI 初始化通信世界。接着，程序会通过读取设备信息和基准测试设置来创建输入和输出缓冲区，并通过RDMA将输入数据提交给执行器。程序的主要逻辑就是循环执行基准测试，将输入数据提交给执行器，并在执行器完成功能调用后获取输出数据，同时测量一些相关时延指标。最后程序通过MPI进行同步和汇总数据，并输出测试结果。

rFaaS是基于RDMA的远程函数服务平台。它提供了一种高效和可扩展的方式来远程执行函数、传输数据和管理资源，使用者可以自由地定义和组合函数。rFaaS框架主要由以下几个部分组成：

- rdma：RDMA的一些实现。
- rfaas：实现了远程函数执行的库。
- executor_manager：用于管理执行程序的HTTP服务。
- resource_manager：用于管理资源的HTTP服务。
- executor：用于在远程机器上执行函数的HTTP服务。
- examples：rFaaS用例程序。
- benchmarks：rFaaS基准测试程序。

下面是rFaaS框架中各个文件的功能列表：

文件名	功能
rFaas/tests/config.h	测试程序配置文件
rFaas/server/atomicops.h	原子性操作库
rFaas/server/common/readerwriterqueue.h	读写队列库
rFaas/rdmalib/lib/rdmalib.cpp	RDMA库实现
rFaas/rdmalib/lib/connection.cpp	建立连接库实现
rFaas/rdmalib/lib/server.cpp	服务器库实现
rFaas/rdmalib/lib/buffer.cpp	缓存库实现
rFaas/rdmalib/lib/util.cpp	工具库实现
rFaas/rdmalib/lib/functions.cpp	函数库实现
rFaas/tests/basic_allocation_test.cpp	基本分配测试代码
rFaas/build/CMakeFiles/3.26.2/CompilerIdCXX/CMakeCXXCompilerId.cpp	编译器ID
rFaas/rfaas/lib/resources.cpp	RFAAS资源库实现

文件名	功能
rFaas/rfaas/lib/connection.cpp	RFAAS连接库实现
rFaas/rfaas/lib/executor.cpp	执行库实现
rFaas/rfaas/lib/devices.cpp	设备库实现
rFaas/examples/functions.cpp	用例函数实现
rFaas/examples/thumbnailer/client.cpp	缩略图客户端代码
rFaas/examples/thumbnailer/opts.cpp	缩略图参数选项
rFaas/examples/thumbnailer/functions.cpp	用于生成缩略图的函数
rFaas/examples/image-recognition/client.cpp	图像识别客户端代码
rFaas/examples/image-recognition/opts.cpp	图像识别参数选项
rFaas/examples/image-recognition/functions.cpp	图像识别函数实现
rFaas/server/executor_manager/settings.cpp	管理器配置代码
rFaas/server/executor_manager/client.cpp	管理器客户端实现
rFaas/server/executor_manager/executor_process.cpp	执行进程实现
rFaas/server/executor_manager/manager.cpp	管理器实现
rFaas/server/executor_manager/opts.cpp	管理器选项
rFaas/server/executor_manager/cli.cpp	管理器命令行选项
rFaas/server/resource_manager/settings.cpp	资源管理器配置代码
rFaas/server/resource_manager/http.cpp	资源管理器HTTP库实现
rFaas/server/resource_manager/manager.cpp	资源管理器实现
rFaas/server/resource_manager/db.cpp	资源管理器数据库实现
rFaas/server/resource_manager/opts.cpp	资源管理器选项
rFaas/server/resource_manager/cli.cpp	资源管理器命令行选项
rFaas/server/executor/server.cpp	执行服务器实现
rFaas/server/executor/opts.cpp	执行服务器选项
rFaas/server/executor/fast_executor.cpp	高速执行器实现

文件名	功能
rFaas/server/executor/functions.cpp	执行服务器函数实现
rFaas/server/executor/cli.cpp	执行服务器命令行选项
rFaas/benchmarks/settings.cpp	基准测试配置代码
rFaas/benchmarks/cpp_interface_opts.cpp	C++接口选项参数解析
rFaas/benchmarks/parallel_invocations.cpp	并行执行测试程序
rFaas/benchmarks/cold_benchmark.cpp	冷测试程序
rFaas/benchmarks/warm_benchmark.cpp	热测试程序
rFaas/benchmarks/cpp_interface.cpp	C++接口测试程序
rFaas/benchmarks/warm_benchmark_opts.cpp	热测试选项参数解析
rFaas/benchmarks/cold_benchmark_opts.cpp	冷测试选项参数解析
rFaas/benchmarks/parallel_invocations_opts.cpp	并发测试选项参数解析
rFaas/benchmarks/scalable_benchmark_opts.cpp	可伸缩性测试选项参数解析
rFaas/benchmarks/scalable_benchmark.cpp	可伸缩性测试程序
rFaas/benchmarks/credentials.cpp	证书库实现
rFaas/build/CMakeFiles/3.26.2/CompilerIdC/CMakeCCompilerId.c	编译器ID