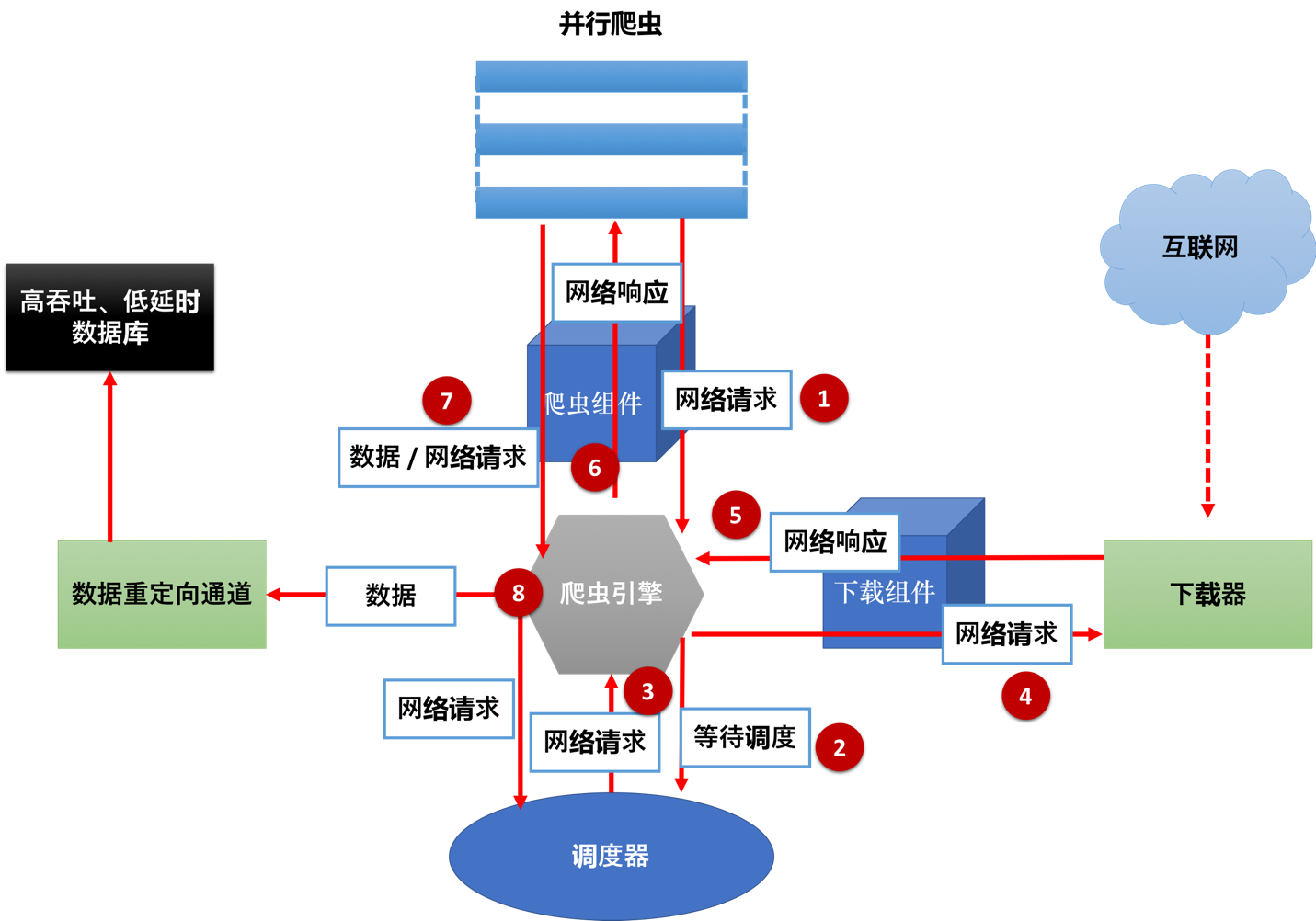
一、爬虫系统架构简介：



上图显示了MQD爬虫系统架构，各模块间业务逻辑、数据流在组件间传递过程（使用红色箭头表示）。

MQD爬虫架构具有如下优势：

(1) 分布式：通过构建异步架构、使用高吞吐低延时非关系型数据库，爬虫系统能够在多台机器上分布式执行。

(2) 可伸缩性：分布式架构，能够通过增加额外的机器和带宽来提高抓取速度。

(3) 性能和有效性：异步架构，能够有效使用各种系统资源，例如，处理器、存储空间和网络带宽。

(4) 可扩展性：爬虫架构设计成模块化的形式，能够支持新的数据格式和新的抓取协议。

MQD异步架构：

整个爬虫系统可以在一个或多个节点上运行。多节点并行抓取的分布式系统需要实现节点之间的通信和调度，在一个爬虫节点上实现并行抓取，需要实现异步 I/O。

MQD采用最先进的前摄型(Proactive)异步架构，爬虫代码主动投递异步操作而不管I/O 设备当前是否可读或可写。投递的异步I/O 操作被系统接管，应用代码也并不阻塞在该操作上，而是指定一个回调函数并继续自己的应用逻辑。当该异步操作完成时，系统将发起通知并调用应用代码指定的回调函数。在前摄式模型中，程序逻辑由各个回调函数串联起来：异步操作A的回调发起异步操作B，B的回调再发起异步操作C，以此往复。

通过异步架构和非关系型数据库，MQD爬虫在内存2GB的云节点上可以实现对3亿条URL 的爬取。

二、数据流

数据流的核心控制器是爬虫引擎，包含如下9个步骤：

1. 引擎接收到爬虫实例的网络请求
2. 引擎将接收到的网络请求转发给调度器，同时从调度器中询问下一个要被执行的网络请求
3. 调度器对当前队列中的网络请求调整优先级，返回给引擎当前优先级最高、即将被执行的网络请求
4. 引擎通过下载组件对网络请求进行一系列处理，最终将请求发送给下载器
5. 当下载器收到远程互联网终端网络响应后，通过下载组件对网络响应进行一系列处理，并将网络响应返回给引擎
6. 引擎接收下载器传回的网络响应，并通过爬虫组件的一系列处理，将网络响应结果返回给爬虫实例
7. 爬虫实例对接收到的网络响应进行解析，并通过爬虫组件的一系列处理，向引擎传送解析后的网络数据、或者新的网络请求
8. 引擎将解析后的数据发送给数据重定向通道（本系统使用内存数据库对数据进行缓存，使用非关系型数据库在硬盘进行序列化存储）；将新的网络请求发送给调度器、并询问下一个网络请求
9. 返回步骤1，不断重复直到调度器中网络请求队列为空

三、模块简介

1. 爬虫引擎

爬虫引擎负责对各个模块间的数据流进行定向，并且当某些特定事件发生时执行相应回调响应函数。

1. 调度器

调度器接收爬虫引擎传入的网络请求，将网络请求按照自定义的规则进行排序，并在引擎查询待执行请求时，时返回当前优先级最高的请求。

1. 下载器

下载器负责网络通信，并将远程服务器响应结果传回爬虫引擎进行下一步处理。

1. 爬虫实例

爬虫实例对网页爬取的具体逻辑进行定义。它们负责解析网络响应、从响应种提取相关数据，并根据网络响应生成新的网络请求。

1. 数据流重定向

数据流重定向负责对爬虫返回的解析结果进行处理。例如对数据进行清理、验证、持久化（内存数据库、硬盘数据库等）处理。

1. 下载组价

下载组件处在爬虫引擎和下载器中间，负责对引擎发送的网络请求，和下载器返回的网络响应进行二次处理。

1. 爬虫组价

爬虫组件处在爬虫实例和爬虫引擎中间，并对爬虫实例的输入（网络响应）和输出（数据、网络请求）进行二次处理。

四、事件驱动模型

本爬虫系统使用事件驱动软件架构，通过异步非阻塞架构实现对互联网的大规模并行访问。