采集系统待完成的开发内容

Beyebe项目·小蜜蜂

功能性部分

以下内容为增强采集系统业务上的功能，每一个都是为了解决一些必须要解决的问题而设计。

以下功能**不按顺序**排列。

采集结果增加Referer属性

在某些情况下需要看这个，在Beyebe.BCrawler.BuildFruit()内填充该属性。

Fruit增加状态：“Json解析失败”、“Jsonp解析失败”、“JS执行异常”

增加这三种状态，并在适当位置填充状态属性，最后在日志显示新增的情况。

页面模型的Content-Type增加jsonp

现在采集器只能处理html和json，需要增加个jsonp。

采集报告发送时机不对

目前采集器每两分钟会向操控中心发送一个当前的采集情况的报告。
但是发送的时机的时机不是很科学，如果采集器命中缓存的话，就不进那个流程，是不会报告的，这就导致在操控中心半天等不到报告，而且采集器管理里面也会看到长时间没响应，以为采集器挂了，体验不好。

重构数据存储方式，以页面模型为单位分数据集

目前的数据存放在MongoDB中，按每个页面url中的host作为数据集名称来分隔存储，不同类型的页面模型没有相关性。

也就是说，一个数据集中会存放很多种页面类型的数据，这对查询非常不友好，因为所有的查询都是查询指定页面模型下的数据，没有跨页面模型查询的场景。

多种页面模型的数据存放在一个数据集中的另一个巨大问题就是索引，如果为了所有页面模型的数据都易查，那么就需要针对各种页面模型建立不同的索引，这会使得数据集中建立太多索引导致插入效率急剧下滑。

另一个问题就是每个数据集都特别巨大，这件事本身对查询也非常不友好。

修改方案是以页面模型的Key作为数据集名称来分隔存储，一个数据集只存一种页面模型的数据。

页面模型增加属性：是否需要抽取数据

> 这件事比较不紧急

在当前的设计中，每一种页面模型都会被抽取数据，并将抽取到的数据Dump到数据中心。

但是存在一种场景，需要对某种页面做特殊操作，但是无需对该页面做数据抽取。

例如，采集企业信息，会在天眼查网站的搜索框中数据某公司的名称进行搜索，因为该网站是模糊搜索，所以搜索结果中会出现许多不正确的公司名称，真正正确的公司名称可能排在搜索结果的中间，而不是第一个。

这种情况目前的做法是在搜索结果页使用js做处理，使用js将需要采集的公司结果移动到搜索结果的第一个（采集路径配置的是采集第一个结果）；或者将其它不正确的结果删除；再或者使用js做click，直接让页面进入到正确的企业详情页。

那么，精确地在搜索结果页做各种操作需要在页面模型中设置，所以需要为搜索结果页添加一个页面模型，来应对这个事情。

从上面的场景中可以看出，搜索结果页并无我们关心的数据，不需要抽取数据，但是这个页面由于映射了相应的页面模型，蜘蛛又会对其抽取数据并Dump到数据中心，所以会导致数据中心有大量该种页面模型的空数据，浪费资源。

解决办法是在页面模型中增加属性，使得可以在界面中设置该页面模型是否需要被抽取数据，为false的页面模型在配置界面不显示“模型数据”标签页。

增加线路管理和任务组并发限制

对采集任务设置“并发组”属性，“并发组”相同的采集任务，采集的是同一网站，某些网站不允许大并发时，可以限制各“并发组”任务的启动数，然后单个任务做线程数控制，以此控制并发。

做这个功能除了对采集任务设置“并发组”，还要对采集服务器设置“线路组”，“线路组”相同的服务器属于同一条宽带。

最终通过配置，能够实现某条线上对某网站的并发上限控制。

对不做数据抽取的页面地址进行采样

为了保证数据的全和准，同时尽量避免爬取不需要的页面，就必须让蜘蛛完全按照期望去运行。

对于需要做数据抽取的页面，有各种日志可以了解其处理情况，但是对于不需要做数据抽取的页面，目前没有相关日志可以观察，所以需要对无页面模型命中，仅对其做url提取，不做数据抽取的页面url做保存。

由于这种url会非常多，所以要做采样，在最多只存100个url的前提下尽量采集多种形状的url。

页面模型的测试结果增强显示

当前的页面模型，点击测试之后只显示catch到的页面中抽取到的数据。如果采集过程中发生了页面跳转等超出预期的情况，测试结果展示区会呈现空。

应当在未catch到页面时，展示浏览器下载到的CrawlStepResult对象，用于调试非预期情况。

采集报告需要重构

因为前段时间爆改采集流程，导致对各计数项执行++操作的位置发生了变化，现在需要对这件事完全重新设计。

删除当前所有++动作，然后重新规划在合理的地方插入++动作。

非功能性需求部分

由于以下功能开发后至少需要三到七天的时间观察和沉淀才能保证其稳定有效，当前在硬件和代码上都没有实验室环境和生产环境的区分，所以在当前项目进度优先的情况下，非功能性需求暂时延后。

以下功能以紧急程度**不按顺序排列**。

重构数据进出服务

之前的数据输入输出服务都由Bee Hive承担，但是后来由于外网的数据需要下载到本地，另单独写了一个下载器，下载器的数据直接对到数据库，这就使得有Bee Hive和下载器两个程序都有数据Dump角色，这不太合理，当数据Dump规则发生改变的时候，需要同时修改两个程序。

在采集量极大时，数据接收只有一个点可能会成为瓶颈，系统在后续的扩展中，可能会使用多MongoDB实例的集群来做支撑，另外还需要使用多个“数据接收服务”和多个“数据输出服务”（如果Hadoop体系稳定，数据输出可以不需要）来增加吞吐和可用性。

所以，在考虑吞吐和可用性方面，Bee Hive也不再适合在作为数据接收服务的同时还承担数据输出服务，读写分离更合理，也更机动。

而且，Bee Hive在开发之初也没有考虑到复杂的数据Dump逻辑，当前的代码有些临时。

当前：

Bee Hive在外网有一个，负责接收所有外网采集器的数据，同时为本地的下载器提供数据输出服务。

Bee Hive在内网有两个，一个只负责接收数据后Dump到多个地方，一个只提供Mongodb订阅数据输出服务。

另有一个数据下载器从外网的Bee Hive获取数据后Dump到多个地方。

重构后：

Bee Hive只负责对外输出Mongidb订阅数据。

另增一种Datahub.Receiver.exe，只用于接收数据，内部使用观察者模式，新增一种数据输出要求（例如数据还要额外推到BBase）时就定制一种观察者，当前的代码是线性的，先Mongodb1，再Mongodb2，再Kafka，这种逻辑一旦在某个环节bug，后续环节就断了，不够稳固。使用互不干涉的观察者模式更为合理。

下载器依然负责将外网的数据下载到本地，但是不再负责数据Dump操作，而是和采集器一样，把数据发送到Datahub.Receiver.exe即可。

重构CacheHub

当前的CacheHub因为没有过多考虑，开发得非常“临时”，需要重构后支持更多环境使用。

当前：

```
1.  //静态的初始化，所有数据使用数据库“db0”
2.  static bool Init(string connectionString)
3.
4.  DateTime? Get(string key);
5.  bool Set(string key);
```

重构后：

```
1.  //非静态，需要指定数据库序号
2.  bool Init(int dbIndex,string connectionString)
3.
```

```
bool Set(string applicationToken,string key);
string Get(string applicationToken,string key);
//有key，且正常删除，true，删除失败，false；无key，返回null。
bool? Delete(string key);

T Get<T>(string key) where T :struct;
```

当前的Get返回的是 `DateTime?`，局限性非常大，换成 `string`，能扩展使用场景。

当前所有数据全部使用db0，在多应用程序场景中，可能key会冲突造成脏数据。重构后需要指定数据库序号，由于不同的数据库key是隔离的，所以能保证数据安全。

对不再使用的第三方进程设置超时关闭

采集器使用了第三方浏览器，因为许多不稳定因素导致这些浏览器进程不能按照预期退出，长时间之后会在操作系统上累计大量无效进程，浪费资源。

目前，第三方进程有两个，都是浏览器进程，分别是 `Chrome` 对象 和 `PhantomJs`对象，这两个对象都继承自 `SeleniumDriver`，`SeleniumDriver.ProcessId`可以获取到进程ID。

正常情况下，进程会在Beyebe.BCrawler.Crawler.ActionBlock.Func.using(browser)块结束后退出。

针对非正常情况，为了实现第三方进程设置超时关闭，将会增加如下动作。

1. 在Beyebe.BCrawler.Crawler.BuildBrowser()中将进程Id `Set(计算机名+进程ID，当前时间)`到CacheHub。
2. 在Beyebe.BCrawler.Client.NormalClient.GetOrigin()内，获取所有当前计算机上的“chromedriver.exe”和“phantomjs.exe”进程，遍历，从CacheHub上`Get(计算机名+进程ID)`，检查进程的生成时间是否超过30分钟，如果超过，关闭进程，并在CacheHub上 `Delete(计算机名+进程ID)`。

注：由于一台机器上可能有多个采集客户端，这件事要考虑多个客户端对单个进程处理的并发。

进程守护

整套系统中有许多种类型的程序，目前的所有“服务”都是以“控制台应用程序”的形式存在的，这些程序有crash的情况，考虑到不管如何提高其稳定性，只要存在更新迭代，不稳定因素会持续存在，所以不能全部寄希望于程序本身保证其稳定可用，最终决定使用进程守护的形式来保证服务稳定可用。

进程守护程序依赖于一套坚固的进程通信组件，所以先开发一个稳固的进程通信组件是前提。

有了进程通信组件之后，被守护程序通过向守护进程发送消息来通知自己的状态，守护进程在发现被守护进程长时间无消息时会kill其进程并重启；在发现进程不存在是，启动其进程。

注意：
进程守护程序需要允许同时在一台服务器上守护多个程序，每个程序设置独立的超时时间和启动参数。

被守护进程可能处于“自动更新中”、“维护中”等非工作状态使得不能及时发送存活消息，要考虑合法的非工作状态如何处理。

需要允许动态编辑被守护进程的清单。

处理Remoting服务不稳定的情况

1. 现在没有断线重连的功能，有些采集器会自己切IP，或者运营商自动变IP，这会影响Remoting连接。
2. 所有Remoting服务端都会因该异常崩溃：Process is terminated due to StackOverflowException.

养蜂人

养蜂人用于动态决定每台服务器上启动多少个采集客户端，所以需要实时监控当前服务器的性能占用。可以考虑将“养蜂人”和“采集客户端的进程守护”合并开发为一个程序。

目前的采集机器分为三种，一种是性能比较强的服务器，一种是闲置的PC机，一种是外网购置

的虚拟机。

为了及时响应某些需要“立即开始”的采集任务，服务器上应当至少预留一个“空转”的采集客户端，专门应对那些需要“立即开始”的采集任务。

目前的服务器上多开采集客户端的方式是将程序复制多份（每份分配了不同的ClientName），分别手动启动，在有养蜂人之后，可以考虑只有一份采集器程序，但是要处理好ClientName的问题。

代码写得丑

主要体现在两点：
1. 有几个对象在某几个点上责任划分得不是很精确。例如采集过程中会发送报告，之前设计的是只有一个对象专门干这事，但是现在这个事情可能有几个对象都要有这个功能，现在走得有点不清不楚。
2. 流程采集器的代码流程（不是功能流程）可以再重构以下，例如有些地方可能用个do while之后有几行代码可能不用写两遍，毕竟跟其它事情相比优先级太低，没有专门的时间没必要轻举妄动。

采集种子的配置需要防呆

当“允许抽取的Url”不为空时，“允许爬行的Url”应该也不允许为空，否则不允许发布。