<HoneyComb >

软件架构文档

版本 <1.0>

[

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <日/月/年> | <x.x> | <详细信息> | <姓名> |
| 2018.9.6 | 1.0 | 初步数据整理和文档编写 | 曹金坤 |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 2

1.1 目的 2

1.2 参考资料 2

2. 用例视图 2

3. 逻辑视图 2

3.1 概述 2

3.2 在构架方面具有重要意义的设计包 2

4. 进程视图 2

5. 部署视图 2

6. 实现视图 2

7. 数据视图（可选） 2

8. 核心算法设计（可选） 2

软件架构文档 （简化版）

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要设计和决策。对于系统架构的描述，会从不同的角度进行说明。

## 参考资料

《HoneyComb项目立项建议书》*,2018.6.20,Honeycomb小组*

《HoneyComb需求规约》*,2018.6.20,Honeycomb小组*

# 用例视图

[本节列出用例模型中的一些用例或场景，这些用例或场景应体现最终系统中重要的、核心的功能；或是在构架方面涉及范围很广（使用了许多构架元素）；或强调或阐明了构架的某一具体的细微之处。]

# 逻辑视图

[本节说明设计模型在构架方面具有重要意义的部分，例如设计模型被分解为多个子系统和包。而每个重要的包又被分解为多个类和类实用程序。您应该介绍那些在构架方面具有重要意义的类，并说明它们的职责，以及几项非常重要的关系、操作和属性。]

## 概述

[本节按照设计模型中包的层次结构来说明设计模型的整体分解情况。]

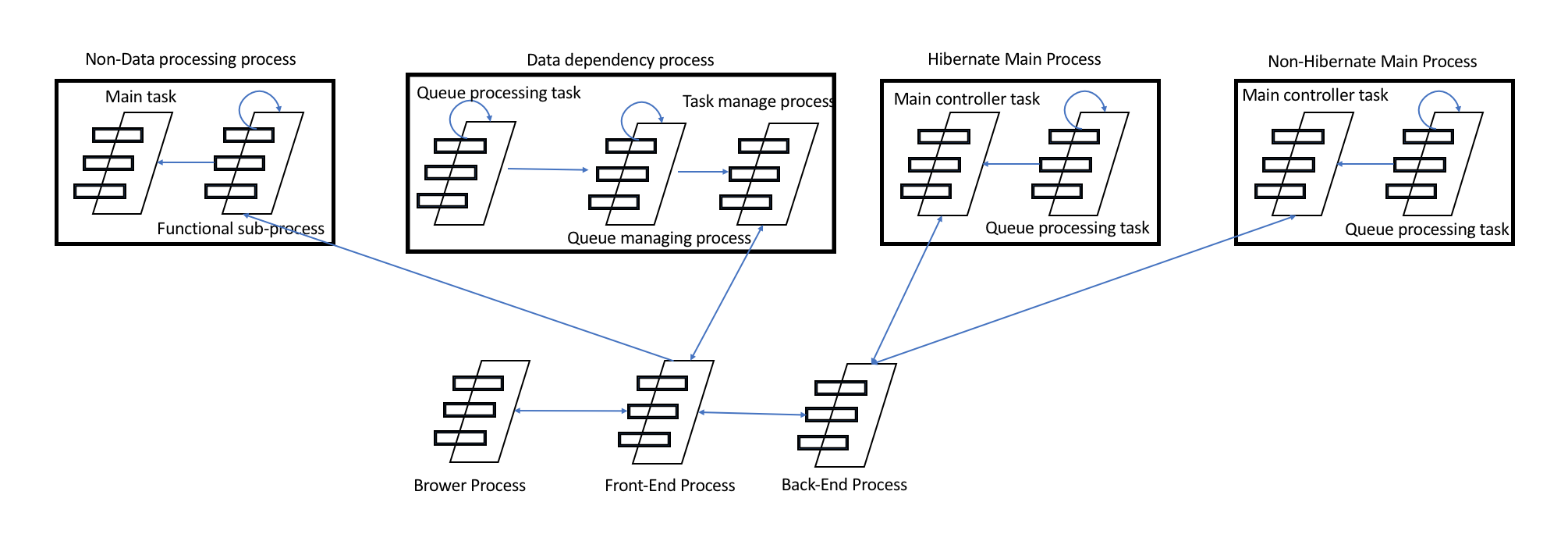
## 在构架方面具有重要意义的设计包

[对于每个重要的包，都用一个小节来加以说明，其中应包括该包的名称、简要说明以及显示该包中所有重要的类和包的图。

对于该包中的每个重要类，应包括其名称、简要说明，还可选择包括对其部分主要职责、操作和属性的说明。]

# 进程视图

[本节说明将系统分解为轻量级进程（单个控制线程）和重量级进程（成组的轻量级进程）的情况。本节的内容按照各个通信或交互的进程组来进行组织。说明进程之间的主要通信模式，例如消息传递、中断和会合。]



功能进程视图示意图

项目的进程设置以三层逻辑为主，体现在两个分离的前后端工程中，三层逻辑分别是：

1. 面向用户的浏览器层对应的以可视化和交互响应为主要工作的进程；
2. 链接后端和浏览器层的前端工程衍生出来的进程组，主要包括两类重进程组，一类是需要对后端发送请求的进程，这类进程中依赖axios进行前后端的通信，另一类是不对后端有请求的纯前端进程，此类进程只需要进行基于Vue的处理即可
3. 后端工程衍生出来的进程类型也有两类，一类是基于Hibernate的对接关系型数据库Mysql的进程组，基于Struts定义的类controller模式；并在进程组中不断使用子进程进行数据处理；另一类是对接Mongodb的进程组，前端接口的controller定义依旧在struts中完成，但是进行工作时，对子进程的调度和Hibernate不同。

# 部署视图

[本节说明用来部署和运行该软件的一种或多种物理网络（硬件）配置。对于每种配置，它至少应该指出执行该软件的物理节点（计算机、CPU）及其互连情况（总线连接、LAN 连接、点到点连接等）。另外还要包括**进程视图**中的各进程到物理节点的映射。]

# 实现视图

[本节说明实现模型的整体结构、软件分解为实现模型中的层和子系统的情况，以及所有在构架方面具有重要意义的构件。]

软件的实现层次中，主要包括了若干重要的子系统和层结构，包括：

1. 数据层：保存了Section 7中描述的所有永久性数据和在使用软件过程中产生的非永久性的持续性和非持续性数据，对于永久性数据的存储采用Mysql和Mongodb两套解决方案并用的设计方法，满足对于结构化数据和非结构化数据的不同需求；
2. 算法层：在系统功能的实现过程中，此层的算法层面承担了从存储的原始数据中分析和提取信息，该层由后端程序中实现，算法的部分主要设计对于数据层保持的数据的操作，主要是查找、比对和返回操作，其中最重要的自实现算法包括比对文本相似度的TF-IDF算法和支持论文网络图构建的MapReduce算法；
3. Hibertnate数据操作系统：基于Hibernate实现，对Mysql数据库进行操作，通过在struts中定义的接口函数，返回需要的结果，一般返回形式为json；
4. Mongodb数据操作系统：对Mongodb数据库进行操作，返回json类型的结果数据；
5. “社区”模块：系统中用于用户分享和交流的子系统，主要的功能涵盖创建板块、发帖、跟帖、添加评论等。该模块中还集成了其他粒度更小的模块，包括编辑器模块、帖子模块等；
6. 论文网络图模块：系统的主体功能部分，主要成分包括搜索论文后扩散开形成的网络图、通过网络图的节点进行论文简介查看、选择变动网络的规模等等；
7. 其他的前端模块：其他的不独立提供主页入口的小模块，包括用户的登录注册模块、网站使用的指导页面等等。

# 数据视图（可选）

[从永久性数据存储方面来对系统进行说明。如果几乎或根本没有永久性数据，或者设计模型与数据模型之间的转换并不重要，那么本节就为可选。]

系统架构中，涉及到的主要永久性数据有：

1. 论文数据

作为Honeycomb项目的主要功能，对于学术论文数据的查找、浏览和可视化都依赖于我们小组事先爬取的公开论文数据。数据来自于对IEEE Xplore网站论文数据的爬取，总数量超过30W条，数据的内容包括：

1. 论文标题
2. 论文作者名单
3. 论文发布的会议/期刊/网站
4. 论文发表年份
5. 论文的简述(Abstract)
6. 论文引用的其他论文

论文数据存储在Mysql数据库中，因为要表示论文之间的“引用”关系、论文和作者之间的“著作”关系，还产生了额外的数据表。

1. 用户登录数据

用户在注册之后，即开始产生永久性数据内容。相关数据在用户自己进行修改的时候才会发生改变，具体的数据条目有：

1. 用户名
2. 用户登录密码
3. 用户个人简介(bio)
4. 用户自定义数据

用户在登录之后，可以在自己的个人空间之中，定制自己的论文收藏和关系网络图，对应的数据为：

1. “社区”板块的后台存储数据

为了支持用户交流而设置的“社区”板块中，用户可以发帖(panel)，可以在别人创建的帖子中跟帖(post)，也可以评论别人发布的内容乃至于评论别人的评论内容，由此产生了非形式化的大量数据，由Mongodb数据库存储，包括：

1. 发帖板块(panel)，包括创建panel的用户ID、时间、标题、内容、关联的评论或者帖子等
2. 帖子(post)，包括发帖人的ID、时间、内容、评论内容和关联的panel信息等
3. 评论(comment)，包括评论人信息、时间、内容、关联的评论或者帖子信息等

# 核心算法设计（可选）

[对系统中的核心算法进行设计。如果没有什么重要的算法，那么本节就为可选。]

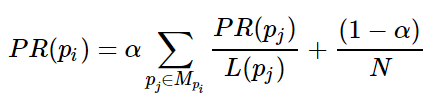
在系统中，主要有两个核心算法参与了主要功能部分，分别是实现搜索关联度匹配的TF-IDF算法和实现数据相似度排序的MapReduce算法。

1. TF-IDF算法：
2. MapReduce算法：

PageRank (简称PR)是一种数学排名算法，常常用于评估网页连接的质量，此评估有助于确定页面重要性和权威的相对分数。在本项目中，我们想要去恰当地衡量每个论文的价值借此挑选出那些值得我们去阅读的好论文而避免在水论文上浪费不必要的时间，我们利用上述的PageRank算法来达成我们这一目标，算法原理首先从以下朴素的想法出发：

1. 当一篇论文被许多论文所引用时，这个论文的价值含量越高
2. 一篇价值含量越高的论文，它所引用的论文的价值含量也会越高，同时论文的作者的知名程度也会越高

PageRank可以建立相应的数学模型来解决这一关系，我们可以先建立一个初步的网络图，网络图中的节点是论文与读者，网络图中的连接（含箭头）代表论文的引用关系，以及撰写关系。

初步的网络关系图确定后，我们为每一个节点赋予相同的起始权重，再根据PageRank来调整不同节点的权重。根据前述原理，定义一个节点的权重值（PR值）如下：

其中PR(pi)是节点的权重值，Mpi是所有引用论文pi的论文集合，L(pj)是论文pj的总引用和总撰写数量之和，N是节点总数，α一般取0.85，α的设置是为了防止数据中的一些论文没有引用对象导致权重值（PR值）无法收敛。

采用以上公式进行权重值的迭代，在不断迭代每个节点的权重值平稳的时候，我们即可获得在已有节点基础上的论文网络图，论文的价值量即可在节点的权重值中体现出来