科技文献知识图谱后端实现

软件工程 2106 王一然

一、任务需求概述

1.条件性搜索：作者、年份、机构单位、期刊名等多个条件搜索，返回对应的论文列表，可参考知网的检索；

2.关键词搜索：关键词的组合搜索，返回论文列表。

二、系统架构

本项目的系统架构采用了分层架构设计，本文详细讨论本人在后端实现的任务需求的内容，其余部分大致说明。各部分架构如下：

2.1 客户端

客户端主要包含 Web 前端，后端通过使用 RESTfulAPI 规范化接口，发送 HTTP 请求与前端服务器进行交互，负责展示数据和接收用户输入。

2.2 服务器

服务器部分是整个系统业务的核心。如有性能方面的更高需求，可考虑将服务器拆分为应用服务器和数据库服务器。

2.2.1 应用服务器

应用服务器端采用 SpringBoot 框架进行开发，负责处理客户端请求、业务逻辑的实现和数据的处理。具体技术选型如下：

SpringBoot：作为应用服务器的框架，提供了快速开发和部署的能力。它通过注解驱动开发，简化了配置，提升了开发效率。

Elasticsearch：用于实现高效的全文检索功能，存储和查询科技文献的相关数据。Elasticsearch 具有强大的搜索和分析能力，能够处理大量数据并提供实时查询的响应。

MyBatis Plus：作为 ORM 框架，简化了数据库操作，提供了丰富的功能和扩

展能力。MyBatis Plus 在 MyBatis 的基础上增加了很多实用的功能，使得数据库访问更加简便和高效。

2.2.2 数据库服务器

数据库服务器主要存储系统中的持久化数据，采用 MySQL 和 Redis 数据库进行存储和管理。

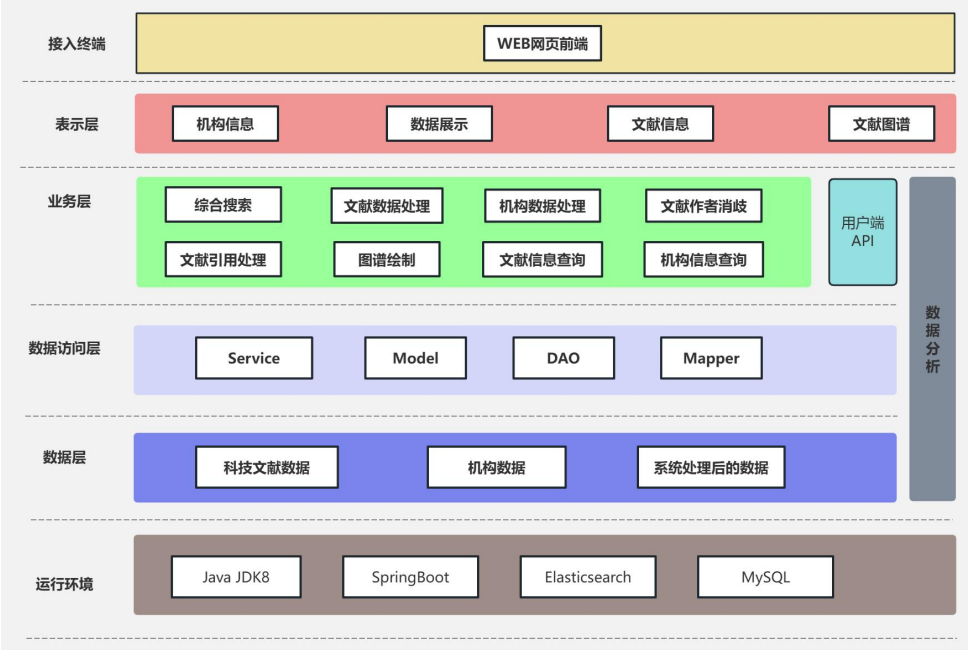
MySQL：关系型数据库，用于存储系统的核心数据，如用户信息、文献信息等。MySQL 提供了丰富的数据管理功能，包括数据查询、更新、删除等操作，并且支持事务处理，保证数据的一致性和完整性。

Redis：作为缓存数据库，用于提高系统的访问速度和响应效率。Redis 以其高性能和支持多种数据结构（如字符串、哈希、列表、集合等），适用于缓存热点数据、会话管理和实时统计等场景。

2.3 系统架构描述

系统架构通过以上技术组件的组合，实现了高效的数据检索、稳定的业务逻辑处理和可靠的数据存储。各组件之间相互协作，形成了一个完整的科技文献知识图谱系统。

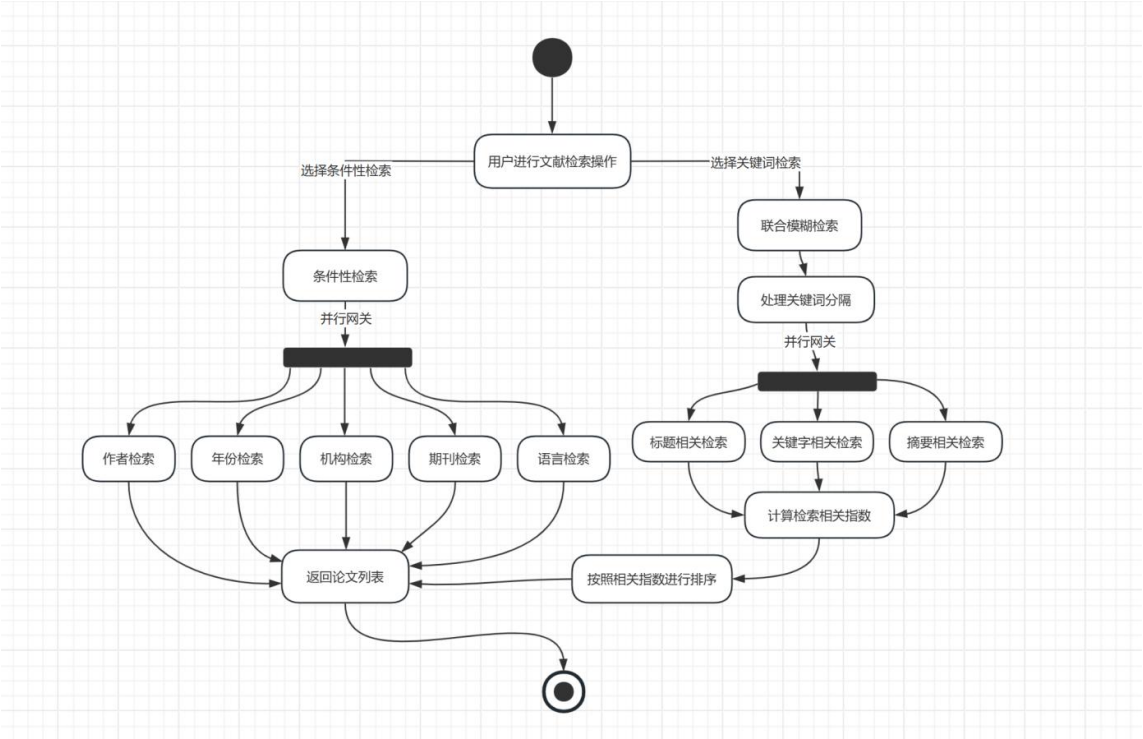
通过以上架构设计，系统能够满足用户对科技文献的高效检索需求，并且具有良好的扩展性和维护性。详细架构图如下所示：



三、功能模块

3.1 论文搜索功能

论文搜索功能主要包括条件性搜索和关键词搜索，用户可以通过多种方式查找所需的文献。项目初期搜索功能逻辑设计如下图所示。



3.1.1 条件性搜索

用户可以通过以下条件进行论文搜索：

作者：根据作者名称进行搜索，返回所有符合条件的论文。

标题：根据文献的标题进行搜索，返回所有与此标题相关的论文。

摘要：根据作者所属机构进行搜索，返回该机构发表的论文。

期刊名：根据期刊名称进行搜索，返回在该期刊上发表的论文。

3.1.2 关键词搜索

用户可以通过输入关键词序列进行联合模糊搜索，系统会在论文的标题、摘要等字段中进行匹配，返回相关的论文列表。同时支持多关键词组合搜索，提高搜索的准确性和相关性。

在实现上，使用 Elasticsearch (ES) 来实现高效的全文检索功能，ES 提供了强大的搜索和分析能力，能够处理大量数据并提供实时查询的响应。

3.2 引用关系处理

引用关系处理主要包括引用关系检索和引用信息解析与查询，通过对文献引用关系的处理，用户可以了解文献间的相互引用情况。

3.2.1 引用关系检索

系统能够检索和处理文献的引用关系，特别是能够处理二跳以内的文献引用关系。用户在查看某篇文献时，系统会展示该文献引用的所有文献，并进一步展示这些被引用文献所引用的文献，帮助用户更全面地了解文献之间的关联。

3.2.2 引用信息解析与查询

系统会解析文献中的引用信息，通过解析引用关系字段（如引用的文章标题、作者、年份等），在 Elasticsearch 中进行查询，找到对应的文献，并返回相关信息。

设计与实现：

在实现上，采用了 MVC 架构模式进行代码的组织和管理，具体分为以下几个层次：

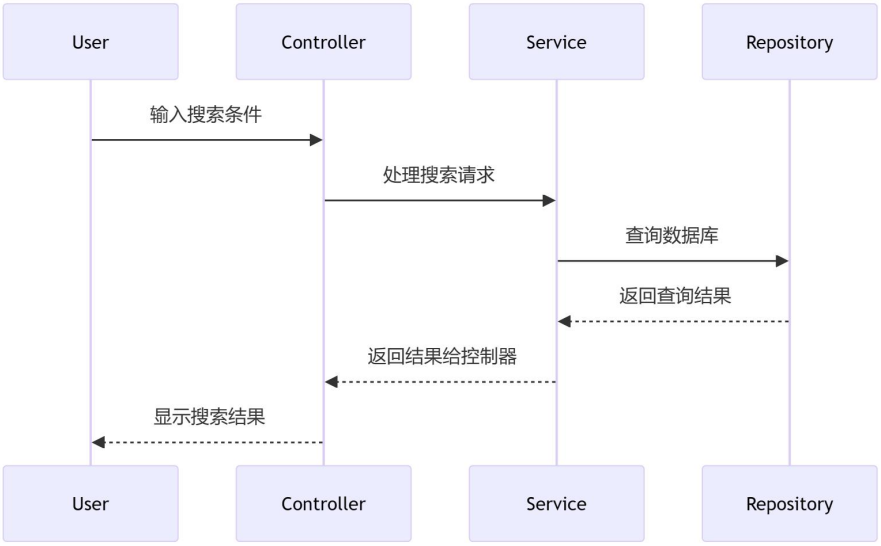
Controller：负责接收客户端请求，并将请求分发给对应的服务层处理。

Service：负责业务逻辑的处理，是应用程序的核心部分。

ServiceImpl：Service 接口的实现类，包含具体的业务逻辑实现。

Mapper：负责数据访问层的操作，与数据库进行直接交互。

在功能设计初期，架构的信息流图如下图所示。



引用关系处理主要通过 Elasticsearch 实现，以下是具体实现方式：

论文查询：使用 Elasticsearch 的多条件查询和模糊查询功能，实现用户对论文的搜索需求。通过组合查询条件（如作者、年份、机构单位、期刊名等）和关键词搜索，返回符合条件的论文列表。

引用关系处理：在检索到的论文中，解析引用信息字段（CR 字段），将引用的文献标题分割并在 Elasticsearch 中进行查询，获取引用的文献信息。对于二跳以内的引用关系，通过递归查询的方式，获取被引用文献的引用信息，并返回完整的引用关系。

通过上述设计，系统能够高效地处理和展示文献的引用关系，并提供多种条件和关键词的组合搜索功能，满足用户对科技文献的高效检索需求。

四、性能优化

为了提高系统的性能，系统在多个方面进行了优化，包括索引优化、查询优化和数据处理优化。

4.1 索引优化

通过合理的索引设计，可以显著提高数据库和搜索引擎的查询性能。

4.1.1 数据库字段索引建立

在 MySQL 数据库中，对频繁查询的字段建立索引。例如，对论文表中的作者、年份、机构单位、期刊名等字段建立索引，以加快查询速度。

在 Elasticsearch 中，对常用的查询字段（如论文标题、摘要、关键词等）设置适当的字段类型和索引方式，例如使用 text 类型进行全文搜索，使用 keyword 类型进行精确匹配。

4.2 查询优化

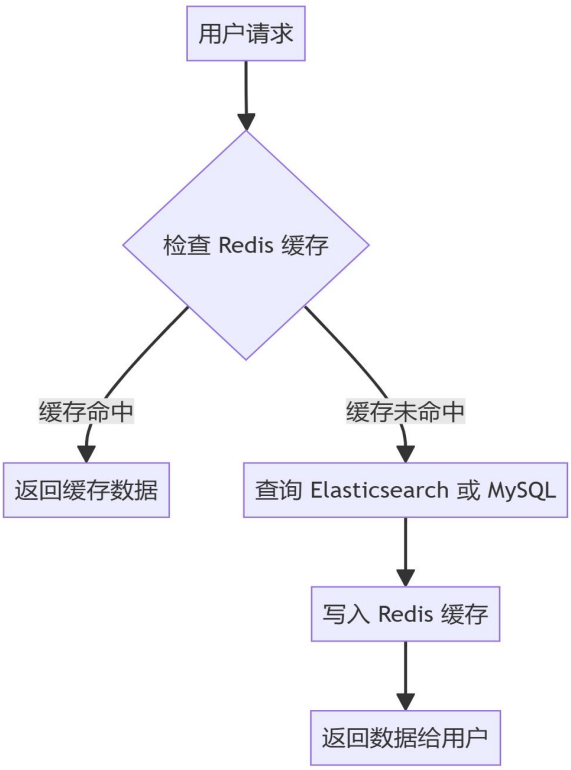
为了提高查询性能，我们采用了 Redis 和 Elasticsearch 进行优化。

4.2.1 使用 Redis 提高查询性能

Redis 作为一个高性能的内存数据库，可以用于缓存频繁访问的数据，减少数据库的压力。

缓存论文信息：在查询论文信息时，先从 Redis 缓存中获取。如果缓存中没有，再从 Elasticsearch 或 MySQL 中查询，并将结果写入缓存。

缓存引用关系：在查询引用关系时，先从 Redis 缓存中获取。如果缓存中没有，再从 Elasticsearch 中查询，并将结果写入缓存。逻辑示意图如下所示。



4.2.2 使用 Elasticsearch 提高查询性能

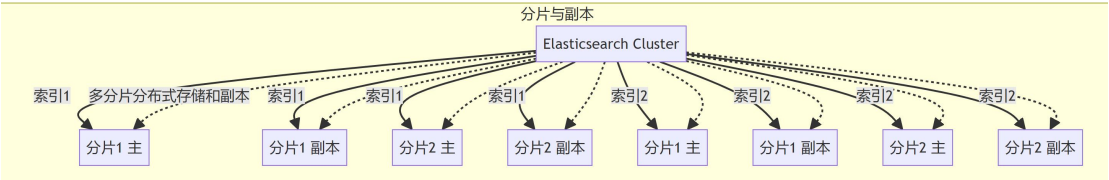
Elasticsearch 作为一个分布式搜索引擎，具有强大的全文搜索和数据分析能

力。我们通过以下方式优化查询性能：

多条件组合查询：在 Elasticsearch 中使用布尔查询（Bool Query）进行多条件组合查询，提高查询效率。

由于 Elasticsearch 可以设置多个节点，因此其具有十分良好的可扩展性，如有性能方面的需求，可以采取分片和副本设置的方式。

分片和副本设置：根据数据量和查询负载，合理设置 Elasticsearch 索引的分片和副本数量，提高数据读取和查询性能。逻辑示意图如下图所示。



4.3 数据处理优化

在数据处理方面，通过将引用的字段写入 Elasticsearch，减少重复查询，提高查询性能。

4.3.1 在查询引用关系后将引用的字段写入 Elasticsearch

由于文献的引用关系是不发生改变的，在检索文献的引用关系后，后端会自动将查询到的引用的文献 ID 列表更新到 Elasticsearch 记录中，这样在后续查询时可以直接获取引用的文献 ID，减少重复查询，大大提升了系统的性能，同时系统中保存的数据也随着用户的使用越来越丰富和完善，真正做到随着使用成长。

通过上述优化措施，我们在索引建立、查询处理和数据处理等方面进行了全面的性能优化，确保系统能够高效处理大规模文献数据，并快速响应用户查询请求。

五、其他说明

5.1 充分使用 Elasticsearch 提供的良好生态

Elasticsearch 提供了非常良好生态。由于该组件已经十分完善，我们可以考虑使用其数据可视化服务组件 Kibana。它能够配合 ES 中存放的数据，产出各类图表和视图，有助于我们分析其中的数据，也能够获得用户操作的数据流。

5.2 未来架构扩展方向

在未来，我们可以考虑与另一个同步开发的项目进行整合，总体采用大数据 Lambda 架构，即将系统分为批处理层、加速层和服务层。在加速层实时进行文献的数据爬取，通过 Kafka 等消息队列组件配合 Spark 快速处理数据，随后存入批处理层进行后续操作。