第二章 系统相关技术

本章节将详细阐述IntelliSearch系统实现过程中所使用到的关键技术。系统基于Qt6开发框架的跨平台能力，结合C++17的现代化特性，通过集成多模态AI服务、构建混合检索架构以及实施高效数据管理，成功实现了一个高性能且具备高度可扩展性的智能搜索引擎系统。接下来本文将从基础框架、核心技术组件及性能优化等多个维度展开并进行深入讨论。

2.2 基础开发框架

本系统在基础开发框架的挑选方面，选定Qt6作为核心框架。做出这一抉择主要综合考量了以下几方面原因。其一，Qt6拥有完善的跨平台解决方案，可与Windows、Linux及macOS等主流操作系统兼容，这极大地拓宽了系统的适用范畴。其二，其Qt Quick技术栈对现代化的声明式UI开发予以支持，有利于快速搭建响应式界面，进而为用户带来顺畅的使用感受。再者，Qt框架具备的信号槽机制与并发工具集，为系统的异步处理给予了强大支撑，特别是在应对复杂的AI服务调用及数据处理任务时，优势尤为明显。

2.3 AI服务集成

在AI服务集成模块的设计实现上，系统设计出统一的服务集成框架，达成了对Kimi、Qwen、Hunyuan以及DeepSeek等大语言模型的顺利接入。通过搭建标准化的服务管理器（AIServiceManager），系统得以对不同AI服务的API调用、响应解析以及错误处理进行统一处置。在并发处理方面，系统运用异步请求模式与智能负载均衡策略，同时借助C++17的std::future和std::async特性，并使之与Qt的事件循环机制协同配合，实现了高效的异步任务处理，由此保证系统在高并发场景下依旧能维持稳定的性能表现。

2.4 检索增强生成技术

检索增强生成（RAG）技术是本系统核心创新点之一。本系统构建起基于RAG的混合检索架构，其中涵盖具备可扩展性的向量检索引擎接口，为后续接入Milvus、Qdrant等向量数据库预留拓展余地。与此同时，系统打造出支持PDF、HTML以及图像等多种格式的文档处理器，并运用流水线架构来提升处理效率。在检索排序环节，系统融合深度学习模型的语义相似度计算与传统的TF - IDF算法，达成了兼顾准确度与效率的混合排序策略。

2.5 数据管理

在数据管理层面，系统运用了多级存储架构的设计思路。选用SQLite作为主体数据库，并借助Qt SQL模块达成高效的数据访问；与此同时，整合Redis作为缓存层级，以此对热点数据的访问性能加以优化。除此之外，系统成功搭建起基于多线程的爬虫系统（CrawlerManager），该系统能够支持数据的增量更新以及智能调度。这种多层次的数据管理架构，一方面保障了数据的持久化存储，另一方面也提供了出色的访问性能。

2.6 性能优化

为保障系统具备高性能与稳定性，本系统于多个维度采取了性能优化举措。在内存管理领域，系统严谨运用智能指针与RAII机制对资源加以管理，从而有效规避了内存泄漏问题；在并发控制方面，借助读写锁和线程池技术对并发访问性能予以优化；在缓存策略层面，构建起涵盖内存缓存及磁盘缓存在内的多级缓存机制，大幅降低了API调用频率。

2.7 监控与日志系统

系统整合了一套完备的监控与日志体系，借助spdlog达成高性能的日志记录工作，且能够支持异步写入以及日志轮转操作。通过精心设计统一的指标收集接口，系统可对关键指标进行实时监控，同时构建起统一的异常处理机制，以此保障系统稳定运行。这些技术综合运用后，让系统在复杂的生产环境里得以维持可靠的性能状态。