제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

KRISTAL-2000 사용자를 위한 JAVA API의 개발

주원균, 이민호, 진두석, 양명석, 정창후, 김광영, 최윤수, 강무영
한국과학기술정보연구원 정보시스템연구실
e-mail:joo@kisti.re.kr

Developments of Java API for KRISTAL-2000

Won-Kyun Joo, Min-Ho Lee, Du-Seok Jin, Myung-Seok Yang,
Chang-Hoo Jung, Kwang-Young Kim, Yun-Soo Choi,
Mu-Yeong Kang
Dept of Information Retrieval,
KISTI(Korea Institute of Science and Technology Information)

요 약

본 논문에서는 트랜잭션 기반의 데이터 관리 및 검색 기능을 갖춘 정보 관리 시스템인 KRISTAL-2000을 간략하게 소개하고, JAVA를 구현언어로 하는 사용자가 해당 시스템의 기능을 원활하게 사용할 수 있도록 하기 위한 JAVA 기반의 KRISTAL-2000 사용자 API 설계를 목표로 한다. 이 때 사용자와 시스템 간의 연결은 텍스트 기반의 소켓 통신을 전제로 한다.

1. 서론

정보 시스템에서 다루는 전문 데이터의 수와 용량이 증가함에 따라 정보 시스템의 각 분야에서는 대용량 데이터의 처리에 초점을 맞춘 연구들이 많이 수행되었는데, 그 중 정보의 저장 및 검색 부분에서는 많은 발전이 있었다.

이런 과정에서 전체 데이터를 여러 개의 DB로 나눈 뒤, 한 시스템 혹은 여러 시스템에 분산 저장하고 이를 통합 검색하는 연구들이 수행되었다. 이런 일련의 연구들 중 하비스트(Harvest)[1]는 초기에 제안되었던 시스템으로, 간단하지만 분산 검색 시스템의 기본적인 구조를 제시하고 이를 직접 설계/구현하여 실례를 보인 시스템으로 유명하다. 또한 에이전트 기반의 분산 검색 모델을 선보인 Harness, 웹을 기반으로 하여 통합 검색 방법을 제시한 여러 메타 검색 엔진들, 프로토콜 기반으로 정보의 융합 검색 방법을 제시한 START[2]등 많은 방법들이 시도되었고, 몇몇 시스템들은 여전히 개발되고 있다. 이런 연구들은 현재 분산 검색, 통합 검색, 분산 통합 검색이라는 용어들로 대표되고 있다.

본 논문에서 대상으로 한 KRISTAL-2000 시스템은 분산 검색의 맥락에서, 대용량 데이터의 분산 검색 기능을 기본으로 제공한다. 또한 그 동안 검색 시스템의 문제점으로 지적되었던 데이터 관리기능을 개선하기 위해, 데이터의 삽입, 수정, 삭제 등의 기능 수행 시 트랜잭션 처리와 회복 기법을 도입하여 제공함으로써, 보다 신뢰성 있는 데이터 관리 기능을 제공한다.

본 논문에서는 분산 정보 관리 시스템 KRISTAL-2000의 기능을 사용자에게 제공하기 위한 JAVA API의 설계와 그 구현에 초점을 맞추었다. 2장에서는 KRISTAL-2000 시스템에 대한 이해를 돕기 위해 시스템의 구조와 각 구성 요소 별 기능에 대해서 설명하고, 3장에서 통신 프로토콜을 기반으로 한 JAVA API의 설계 및 구현 과정에 대해 설명하고 4장에서 결론을 맺는다.

2. KRISTAL-2000 시스템 구조

분산 환경에서 정보의 원활한 관리와 효과적인 검색을 지원하기 위하여 KRISTAL-2000 시스템은 그림 1과 같은 구조로 설계/구현되었다. 시스템 구조에서 두드러진 특징은 중요 구성요소(JOB

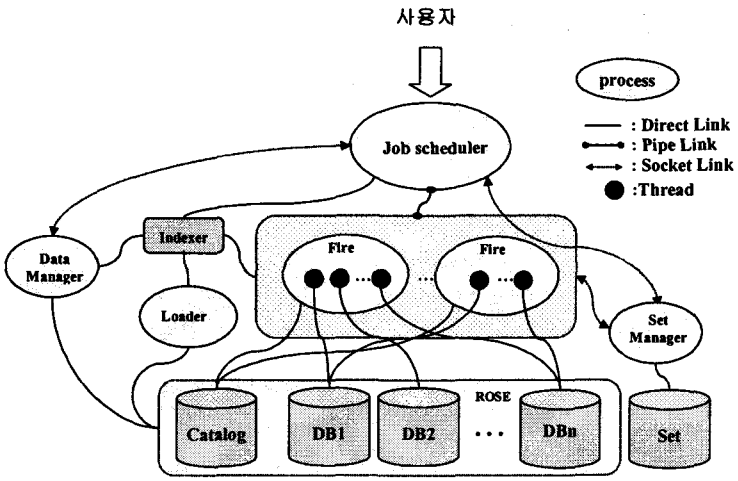

그림 1 KRISTAL-2000 시스템의 구조

Scheduler, Fire, Set Manager, Data Manager)들이 독립적인 데몬 프로세스의 형태를 취하고 있어 별도의 시스템에 이식 가능한 형태로 설계되었다는 점이다. 이해를 돕기 위해 중요 구성요소에 대해서 간략하게 설명한다.

■ 작업 스케줄러(Job Scheduler) : 사용자와 밀접한 연관이 있는 부분으로서 사용자의 요청(온라인 문서 관리/검색)을 받아들여 각 하위 시스템에 전달하여 결과를 얻을 수 있도록 도와주는 부분으로서, 데이터 관리기/검색기/셋 관리기와 상호작용 하에 작업을 수행한다.

■ 데이터 관리기(Data Manager) : 온라인 문서 관리를 주요 기능으로 하는 요소로서 데이터 삽입/삭제/수정의 역할을 수행한다. 기능 수행 시 트랜잭션 처리를 수반한다.

■ 검색기(Fire) : 검색을 담당하는 부분으로서, 불리언과 벡터 모델의 확장된 검색 방법을 제공하고, 셋 관리기와 상호 작용을 함으로써 효율적인 검색을 제공한다. 또한 검색을 위한 포스팅 정보의 효과적인 수집을 위해서 Shore[3] 쓰레드를 사용하여 DB에 연결된다.

■ 셋 관리기(Set Manager) : 검색 결과의 저장 및 관리를 담당하는 부분으로서, 사용자가 로컬에 검색 결과를 저장하지 않고서도 자신의 검색 결과를 탐색 및 관리 할 수 있도록 하고, 캐시(cache) 기능을 제공함으로써 검색기의 빠른 검색 수행을 보조한다.

각 구성 요소들 간의 중요 연결 관계는 그림 1에서 세 가지 선을 이용하여 설명하고 있는데, 각 관계는 동일 프로세스에 존재하거나 파이프(pipe)로 연결된 프로세스 혹은 소켓(socket)으로 연결된 프로세스 형태를 취하고 있다. 이러한 연결 관계는 각 구성 요소의 독립성을 보장한다.

JAVA 사용자 API는 사용자와 작업 스케줄러(이하 서버)를 소켓 기반의 통신 프로토콜을 이용하여 연결한다.

3. KRISTAL-2000 JAVA API 설계

3장에서는 JAVA API를 설계하는 과정을 설명하는데, KRISTAL-2000 API 명세, 통신 프로토콜, 입출력 클래스, 인코딩 방식에 관한 내용을 다룬다.

3.1 KRISTAL-2000 API 명세

KRISTAL-2000 API는 크게 3 그룹으로 구성되는데, 각 그룹 내에 속한 API와 각 API에 대한 설명은 다음에 기술한다.

그룹 1: 에러 처리 관련 API	
CL_GetErrMsg	에러 코드와 메시지 출력

그룹 2: 검색 관련 API	
CL_GetDBList	DB 리스트 구함
CL_GetSectionList	섹션 리스트 정보 구함
CL_Sort	검색결과 정열
CL_Search	확장된 불리언/벡터 검색
CL_ResultSearch	
CL_SimSearch	
CL_GetMetaResult	검색 결과에 대한 정보 구함
CL_GetDocList	검색 결과 문서 리스트 구함
CL_GetSections	문서 내용 구함

그룹 3: 온라인 문서관리 문서 관리 API	
CL_AppendParsedDoc	파싱 된 문서 삽입
CL_UpdateParsedDoc	파싱 된 문서 갱신
CL_DeleteDoc	문서 삭제
CL_AppendBlobSections	
CL_UpdateBlobSections	
CL_GetBlobSections	

3.2 KRISTAL-2000 프로토콜 구조

KRISTAL-2000 프로토콜은 텍스트 형태를 갖으며 헤더와 데이터 부분으로 분리되어 전송되는데, 각각에 대한 패킷 구조는 다음과 같다.

■ 헤더 패킷

DST	SRC	Data Length	MSG Type

DST : 목적지 코드 (예 : FIRE)
SRC : 원본 코드 (예 : SM)
Data Length : 데이터 패킷의 길이
MSG Type : 메시지 타입 코드 (예: FIRE_SEARCH)

■ 데이터 패킷

헤더 패킷을 전송한 후에 바로 데이터 패킷을 전송한다. 데이터 패킷은 클라이언트에서 서버로 전달하느냐 혹은 서버에서 클라이언트로 전송되느냐에 따라 두 가지 종류가 있다.

클라이언트에서 서버로 전달할 때는 순수 데이터 부분만 있으며, 서버에서 클라이언트로 전송될 때는 에러코드(err-code)와 데이터의 두 부분으로 구성된다. 데이터 부분은 요청하는 서비스(메시지 타입) 별로 각각 다르다.

경우 1 : 클라이언트에서 서버로 전송 시

Data(데이터)

경우 2 : 서버에서 클라이언트로 전송 시

ErrCode(에러코드)	Data(데이터)

- 에러코드 : 0 이면 OK, 그 외는 해당 에러코드.
- 데이터: 에러코드가 OK일 경우, 해당 메시지 타입별 결과 데이터를 붙인다. OK 가 아닐 경우에는 에러코드에 대한 에러 메시지가 뒤따른다. 따라서 클라이언트에서는 요청한 서비스에 대해 서버의 어느 부분에서 이상이 발생하였는지 즉각 파악할 수 있다. 에러 코드 값과 에러 메시지에 대한 것은 [4]를 참조하고, 메시지 타입별 데이터 부분의 구조는 경우 1과 경우 2로 나누어 다음 절에서 설명한다.

3.3 KRISTAL-2000 프로토콜 형식

하위 절의 각 표 안의 첫줄은 위의 경우 1에 대한 데이터에 대한 명세이고, 두 번째 줄은 경우 2에 대한 데이터에 대한 명세이다. 이 때 데이터 부분은 텍스트로 구성되는데, 각 필드는 세미콜론으로 구분한다. 프로토콜 형식은 그룹2(검색 관련된 내용)와 그룹 3(온라인 문서 관리 관련)의 두 부분으로 나누어 명세 한다.

3.3.1 검색 관련

다음은 검색에 관련된 9개의 함수에 대한 데이터 패킷 내용이다

■ CL_GetDBList

NULL
개수;(DB이름;문서개수;DB크기;)

■ CL_GetSectionList

NULL
개수;(섹션;색인타입;섹션타입;union개수;(union섹션이름;))

■ CL_Sort

결과셋번호;섹션이름;정열순서
새결과셋번호;개수

■ CL_Search

검색방법;DB리스트;질의길이;질의
결과셋번호;개수

■ CL_ResultSearch

결과셋번호;섹션리스트;질의길이;질의
결과셋번호;개수

■ CL_SimSearch

문서ID;DB리스트;섹션리스트;유사도값;모드
결과셋번호;개수

■ CL_GetMetaResult

결과셋번호
불용어;원질의;원검색방법;DB;확장검색방법;확장DB목록;확장질의크기;확장질의

■ CL_GetDocList

결과셋번호;시작위치;개수;섹션이름
개수;(문서ID;섹션개수;(길이;섹션이름;길이;섹션값))

■ CL_GetSections

문서아이디;섹션개수;(섹션이름)
개수;(길이;섹션이름;길이;섹션값)

3.3.2 온라인 문서 관리 관련

다음은 온라인 문서 관리에 관련된 6개의 함수에 대한 데이터 패킷 내용이다.

■ CL_AppendParsedDoc

DB이름;섹션개수;(길이;섹션이름;길이;섹션값;)인코딩타입
문서ID

■ CL_UpdateParsedDoc

문서ID;섹션개수;(길이;섹션이름;길이;섹션값)
NULL

■ CL_DeleteDoc

문서ID
NULL

■ CL_AppendBlobSections

DB이름;섹션개수;(길이;섹션이름;길이;섹션값)
개수;(길이;섹션이름;길이;섹션값)

■ CL_UpdateBlobSections

문서ID;섹션개수;(길이;섹션이름;길이;섹션값)
NULL

■ CL_GetBlobSections

문서ID;섹션개수;섹션이름
개수;(길이;섹션이름;길이;섹션값)

3.4 입출력 관련 클래스

위에서 정의 된 각 함수와 관련된 입출력 클래스에 대한 내용은 다음과 같다.

```
public class MetaDB {
  public String   DBName;     /* DB 이름 */
  public int Cardinality;     /* 문서 수 */
  public int DBSize;          /* DB 크기 */
}

public class MetaSec {
  public String SecName;      //섹션 이름
  public int SecType;         //섹션 타입
  public String IdxType;      //색인 타입
  public Vector UniSecList;   //union 섹션 리스트
}

public class MetaTerm {
  public String SecName;      //섹션이름
  public String Term;         //단어
}

public class MetaResult {
  public int SearchMethod;    //검색방법
  public int OperationType;
  public Vector DBList;       //DB 리스트
  public String OriginalQuery; //원질의
  public String ExtendQuery;  //확장질의
  public Vector QueryTermList; //질의단어리스트
  public Vector StopwdList;   //불용어리스트
}

public class ResDoc {
  public int DocId;           //문서 아이디
  public double Weight;       //가중치
  public Vector SecList;      //섹션 리스트
}

public class ResDocList {
  public int count;  //검색 문서 수
  public Vector DocList;      //문서 ID 리스트
}

public class ResSec {
  protected String SecName;   //섹션 이름
  protected String SecValue;  //섹션 값
  protected byte[] BinSecValue;//바이너리 섹션 값
}

public class ResSet {
  public int Setnum;          //결과 셋 번호
  public int count;           //결과 문서 수
  public int docid;
}
```

3.5 메시지 인코딩

메시지 인코딩은 KRISTAL-2000 통신 프로토콜에서 사용하는 방법과 JAVA 스트링 객체 내에서 사용하는 방법의 두 가지로 구분하여 설명한다.

■ KRISTAL-2000 통신 프로토콜

내부적으로 바이너리 형태의 데이터 전송 시에 base-64방식의 인코딩을 사용한다.

■ JAVA 스트링 객체

KRISTAL은 내부적으로 모든 데이터를 UTF-8형태로 처리하는 반면, JAVA String 클래스는 자신만의 고유한 Unicode 처리 방식을 따른다. 따라서 JAVA에서 문자열 타입의 처리를 위해 JAVA String 객체를 사용하고자 한다면 JAVA String 객체와 KRISTAL-2000 사이의 데이터 변환이 필요하다.

- JAVA String(Unicode)을 UTF8로 변환

```
String str; //입력 스트링
byte[] tstr = str.getBytes("UTF8");
```

- UTF8을 JAVA 스트링(Unicode)으로 변환

```
byte[] str; //입력 스트림
String val = new String( str, "UTF8" );
```

6. 결론

본 논문에서는 JAVA를 개발 언어로 삼는 사용자(개발자)가 분산 정보 관리시스템인 KRISTAL-2000을 이용하기 위한 JAVA 기반의 사용자 API의 설계 및 구현에 그 목적이 있었다. JAVA 기반의 API는 통신 프로토콜을 매개로 하여 사용자와 서버를 연결시켜주었다. 총 3개의 그룹으로 나누어 각 그룹에 맞는 API를 구현하였다.

참고문헌

[1] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. Harvest: A Scalable, Customizable Discovery and Access System Technical Report CU-CS-732-94, August 26, 1994 Department of Computer Science, University of Colorado - Boulder.

[2] Luis Gravano, Chen-Chuan K.Chang, Hector Garcia-Molina and Andreas Paepcke, "START: Stanford Proposal for Internet Meta-Searching", ACM SIGMOD Record, Vol.26, No.2, Pages 207-218, 1997

[3] http://www.cs.wisc.edu/shore/

[4] KRISTAL 기술 매뉴얼

[5] 주원균, 이민호, 강무영, "분산 정보 관리 시스템을 위한 사용자 API 설계 및 구현", 정보과학회 추계 학술 발표 논문집, 2002