7주차

9장 웹 크롤러 설계

웹크롤러?

검색 엔진에서 널리 쓰는 기술로 웹에 새로 올라오거나 갱신된 콘텐츠를 찾아내는 것이 주된 목적이다.

웹 크롤러 활용

검색 엔진 인덱싱

- 크롤러의 가장 보편적인 용례, 크롤러는 웹 페이지를 모아 검색 엔진을 위한 로컬 인덱스를 만든다. 구글봇은 구글검색 엔진이 사용하는 웹 크롤러이다.

웹 아카이빙

- 나중에 사용할 목적으로 장기 보관하기 위해서 웹에서 정보를 모으는 절차를 말한다.

웹 마이닝

- 웹 마이닝을 통해 인터넷에서 유용한 지식을 도출해낼 수 있다. 유명 금융 기업들은 크롤러를 사용해 주주총회 자료나 연차보고서를 다운 받아 기업의 핵심 사업 방향을 알아낸다

웹 모니터링

- 크롤러를 사용하면 인터넷에서 저작권이나 상표권이 침해되는 사례를 모니터링할 수 있다. 사진 도용방지 같은 경우 디지털워터마킹 같은 것을 사용해서 찾는다.

1단계 - 문제 이해 및 설계 범위 확정

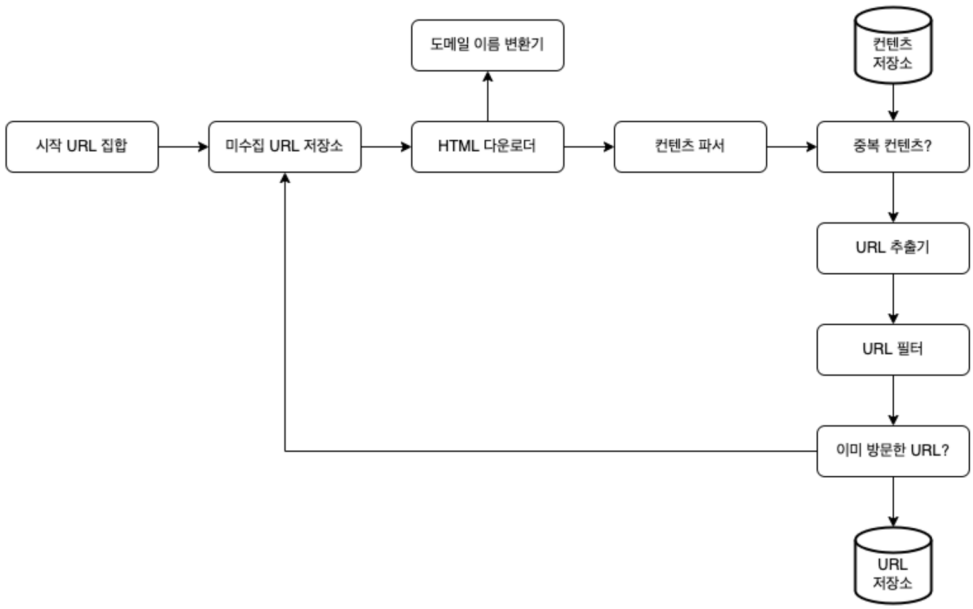
웹 크롤러 알고리즘

1. URL 집합의 모든 웹 페이지를 다운로드한다
2. 다운받은 웹 페이지에서 URL들을 추출한다
3. 추출한 URL을 다운로드할 URL목록에 추가하고 계속 과정1부터 반복한다

속성

- **규모 확장성** : 여러 대의 크롤러로 병렬 수집한다.
- **안정성(robustness)** : 비정상적 입력이나 환경에 대응한다.
- **예절(politeness)** : 짧은 시간에 너무 많은 요청을 보내면 안된다.
- **확장** : 새로운 형태의 콘텐츠(음악, 영상 등)를 쉽게 지원해야 한다.

2단계 - 개략적 설계안 제시 및 동의 구하기



- **시작 URL 집합** : 크롤링을 시작하는 출발점으로, 크롤러가 **최대한 많은 링크를 탐색할 수 있도록** 하는 URL을 고르자
    - 일반적으로, 전체 URL 공간을 작은 부분집합으로 나눈다 ex) 주제별로 URL 공간을 세분화하여 각각 다른 시작 URL 을 사용
- **미수집 URL 저장소** : 다운로드 할 URL을 저장 및 관리하는 컴포넌트 (FIFO 큐)
- **HTML 다운로더** : 인터넷에서 웹 페이지 다운로드하는 컴포넌트
- **도메인 이름 변환기** : 웹페이지를 다운받기 위해 URL을 IP주소로 변환하는 절차를 담당
- **콘텐츠 파서** : 웹 페이지 다운 시 파싱, 검증 절차를 거쳐야 함
- **중복 콘텐츠인가?** : 이미 시스템에 저장된 콘텐츠임을 알아내기 위해 웹 페이지의 해시 값을 비교하여 중복 콘텐츠를 1번만 저장하도록 함
- **콘텐츠 저장소** : HTML 문서 보관하는 시스템
- **URL 추출기** : HTML 페이지를 파싱하여 링크를 골라내는 역할
- **URL 필터** : 특정 콘텐츠 타입/확장자를 갖는 URL을 크롤링 대상에서 배제하는 역할
- **이미 방문한 URL?** : 블룸 필터나 해시테이블로 이미 방문한 URL이나 미수집 URL 을 추적할 수 있게 함
    - 블룸 필터는 원소가 집합에 속하는지 여부를 검사하는데 사용되는 확률적 자료 구조
- **URL 저장소** : 이미 방문한 URL 보관하는 저장소

웹 크롤러 작업 흐름

1. 시작 URL들을 미수집 URL 저장소에 저장한다
2. HTML 다운로더는 미수집 URL 저장소에서 URL 목록을 가져온다
3. HTML 다운로더는 도메인 이름 변환기를 사용하여 URL의 IP주소를 알아내고, 해당 IP 주소로 접속하여 웹 페이지를 다운 받는다
4. 콘텐츠파서는 다운된 HTML 페이지를 파싱하여 올바른 형식을 갖춘 페이지인지 검증한다
5. 콘텐츠 파싱과 검증이 끝나면 중복 콘텐츠인지 확인하는 절차를 개시한다

6. 중복 콘텐츠인지 확인하기 위해서, 해당 페이지가 이미 저장소에 있는 지 본다
- 이미 저장소에 있는 경우에는 처리하지 않고 버린다
- 저장소에 없는 콘텐츠인 경우에는 저장소에 저장한뒤 URL 추출기로 전달한다
7. URL 추출기는 해당 HTML페이지에서 링크를 골라낸다
8. 골라낸 링크를 URL필터로 전달한다
9. 필터링이 끝나고 남은 URL만 중복 URL판별 단계로 전달한다
10. 이미 처리한 URL인지 확인하기 위하여 URL 저장소에 보관된 URL인지 살핀다. 이미 저장소에 있는 URL은 버린다
11. 저장소에 없는 URL은 URL 저장소에 저장할 뿐만아니라 미수집 URL 저장소에도 전달한다

3단계 - 상세설계

가장 중요한 컴포넌트 & 구현 기술을 상세하게 보자

DFS 를 쓸 것인가, BFS 를 쓸것인가

웹은 유향 그래프나 같다. 페이지는 노드, 하이퍼링크는 엣지라고 보면 된다.

크롤링이 그래프를 탐색하는 과정이라고 생각하면 된다

- 웹 그래프의 깊이를 알 수 없으므로 DFS는 적합하지 않다.
- 링크는 같은 서브 도메인을 가리키는 경향이 크기 때문에, BFS(너비 우선 탐색)를 사용하면 한 서버에 많은 부하를 줄 수 있다.
- 그래서 페이지 순위, 트래픽 등을 지표로 BFS 큐의 우선순위를 조정한다.

이 구현법에는 두 가지 문제점이 있다.

1. 한 페이지에서 나오는 링크의 상당수는 같은 서버로 되돌아간다. 결국 크롤러는 같은 호스트에 속한 많은 링크를 다운받느라 바빠지게 되는데, 이때 링크들을 병렬로 처리하게 된다면 수 많은 요청으로 과부하에 걸리게 될 것이며 이런 크롤러는 보통 예의 없는 크롤러로 간주된다.
2. 표준적 BFS 알고리즘은 URL 간에 우선순위를 두지 않는다. 처리 순서에 있어 모든 페이지를 공평하게 대우한다는 뜻이다. 하지만 모든 웹 페이지가 같은 수준의 품질, 같은

> 수준의 중요성을 갖지 않는다. 그러니 페이지 순위, 사용자 트래픽의 양, 업데이트 빈도 등 여러가지 척도에 비추어 처리 우선순위를 구별하는 것이 바람직하다.

미수집 URL 저장소

미수집 URL 저장소를 활용하면 이런 문제를 좀 쉽게 해결할 수 있다. URL 저장소는 다운로드할 URL을 보관하는 장소다. 이 저장소를 잘 구현하면 예의를 갖춘 크롤러, URL 사이의 우선 순위와 신선도를 구별하는 크롤러를 구현할 수 있다.

예의

웹 크롤러는 수집 대상 서버로 짧은 시간 안에 너무 많은 요청을 보내는 것을 삼가야 한다.

예의 바른 크롤러를 만드는 데 있어서 지켜야 할 한 가지 원칙은 동일 웹 사이트에 대해서는 한 번에 한 페이지만 요청한다는 것이다. 이 요구사항을 만족시키려면 웹 사이트의 호스트명과 다운로드를 수행하는 작업 스레드 사이의 관계를 유지하면 된다. 즉 각 다운로드 스레드는 별도의 FIFO 큐를 가지고 있어서 해당 큐에서 꺼낸 URL만 다운로드한다.

- 큐 라우터 : 같은 호스트에 속한 URL은 언제나 같은 큐로 가지도록 보장하는 역할을 한다.
- 매핑 테이블 : 호스트 이름과 큐 사이의 관계를 보관하는 테이블
- FIFO 큐 : 같은 호스트에 속한 URI은 언제나 같은 큐에 보관된다.
- 큐 선택기 : 큐 선택기는 큐들을 순회하면서 큐에서 URL을 꺼내서 해당 큐에서 나온 URL을 다운로드하도록 지정된 작업 스레드에 전달하는 역할을 한다.
- 작업 스레드 : 작업 스레드는 전달된 URL을 다운로드하는 작업을 수행한다. 전달된 URL은 순차적으로 처리될 것이며, 작업들 사이에는 일정한 지연시간을 둘 수 있다.

우선순위

유용성에 따라 URL의 우선순위를 나눌 때는 페이지 랭크, 트래픽 양, 갱신 빈도 등 다양한 척도를 사용할 수 있을 것이다.

본 절에서 설명할 순위결정장치는 URL 우선순위를 정하는 컴포넌트다.

큐에 URL을 저장하기 전에 순위 결정 장치를 거치도록 설계를 변경한다.

1. 순위 결정 장치: URL 을 입력으로 받아 우선순위를 계산.
2. 우선순위 별로 큐가 하나씩 할당된다. 우선순위가 높으면 선택될 확률도 올라간다.

3. 큐 선택기: 임의 큐에서 처리할 URL을 꺼낸다. 순위가 높은 큐에서 더 자주 꺼내도록 프로그램 되어있다.

이 둘을 반영한 전체 설계는 다음과 같다.

- 전면 큐 (front queue): 우선순위 결정 과정을 처리한다.
- 후면 큐 (back queue): 크롤러의 예의를 보장한다.

신선도

웹 페이지는 수시로 변경되므로, 이미 다운로드 했더라도 주기적으로 재수집할 필요성이 있음 하지만, 모든 URL을 재수집하는 것은 많은 시간과 자원이 필요하므로, 웹 페이지 변경 이력을 활용하거나 우선순위를 활용해서 중요 페이지를 자주 재수집한다

HTML 다운로더

HTML 다운로더는 HTTP 프로토콜을 통해 웹페이지를 내려 받음.

Robots.txt

로봇 제외 프로토콜이라고도 불린다. 웹사이트가 크롤러와 소통하는 표준.

이 파일에는 크롤러가 수집해도 되는 페이지 목록이 들어있다. 따라서 크롤러는 사이트를 다운받기 전에 해당 파일에 나열된 규칙을 먼저 확인해야 한다.

같은 호스트에서 Robots.txt 파일을 중복으로 다운로드 하는 것을 피하기 위해, 이 파일은 주기적으로 다시 다운 받아 캐싱한다.

성능 최적화

HTML 다운로더에 사용할 수 있는 성능 최적화 기법들

1. 분산 크롤링 : 성능을 높이기 위해 크롤링 작업을 여러 서버에 분산한다.
2. 도메인 이름 변환 결과 캐시 : DNS resolver(도메인 이름변환기)는 크롤러 성능의 병목 중 하나인데, 이는 DNS 요청을 보내고 결과를 받는 작업의 동기적 특성 때문이다. 스레드는 DNS 요청의 결과를 받기 전까지는 다음 작업을 진행할 수 없다. 크롤러 스레드 가운데 어느 하나라도 DNS resolver에 요청을 보내면, 이 요청이 완료될 때까지 다른 스레드의 요청이 모두 block된다. 따라서 DNS 조회 결과로 얻어진 도메인 이름과 그에 상응하는 IP 주소를 캐시에 보관, 주기적으로 갱신하도록 구현하여 성능을 높일 수 있다.

3. 지역성: 크롤링 작업을 수행하는 서버를 지역별로 분산하는 방법이다. 크롤링 서버가 크롤링 대상 서버와 지역적으로 가까우면 페이지 다운로드 시간이 줄어들것이다. 지역성을 활용하는 전략은 크롤서버, 캐시, 큐, 저장소 등 대부분의 컴포넌트에 적용 가능하다.
4. 짧은 타임아웃: 어떤 웹서버는 응답이 느리거나 아예 응답하지 않는데, 이런 경우를 대비해 크롤러가 최대 얼마나 기다릴지를 미리 정해두는 것이다. 타임아웃의 경우 해당 페이지 다운로드를 중단한다.

안정성

안정성은 성능 최적화 만큼 중요하게 고려해야할 부분이다. 시스템 안정성을 향상시키기 위한 접근법 중 몇가지를 보자.

1. 안정 해시(consistent hashing): 다운로더 서버(분산 HTML 다운로더)들에 부하를 고르게 분산하기 위해 적용가능한 기술이다.
2. 크롤링 상태 및 수집 데이터 저장: 장애가 발생한 경우에도 쉽게 복구할 수 있도록
    - 크롤링 상태
    - 수집된 데이터 를 지속적 저장장치에 기록해두는 것이 바람직하다.
3. 예외처리: 예외가 발생해도 전체 시스템이 중단되지 않도록 미리 처리한다.
4. 데이터 검증: 시스템 오류를 방지하기 위한 중요 수단이다.

확장성

- 기능별로 컴포넌트를 분리해서 새로운 컴포넌트를 추가하고 변경하기 용이하도록 한다.

문제 있는 콘텐츠 감지 및 회피

- 중복 컨텐츠 : 해시나 체크섬으로 탐지
- 거미 덫 : 무한히 깊은 URL에 빠지지 않도록 최대 길이를 제한.
- 데이터 노이즈: 광고, 스팸 URL 등을 제거한다.

4단계 : 마무리

- 좋은 크롤러가 갖추어야 하는 특성 : 규모 확장성, 예의, 확장성, 안정성

- 추가 논의 가능한 사항
    - Server-Side Rendering : 웹 사이트가 링크를 즉석에서 만들어내므로 있는 그대로 다운받아 파싱하면 동적 생성되는 링크를 발견할 수 없음. 따라서 SSR을 적용하면 해결 가능
    - 원치 않는 페이지 필터링 : 크롤링에 소요되는 자원은 한정적이므로, 스팸 방지 컴포넌트로 스팸성 페이지를 걸러내자
    - DB 다중화 및 샤딩 : 이 기법을 적용하면 데이터 계층의 가용성, 규모 확장성, 안정성 향상됨
    - 수평적 규모 확장성 : 대규모 크롤링을 위해서는 많은 서버 필요, 이 때 중요한 것이 서버가 무상태를 유지하도록 하는 것
    - 가용성, 일관성, 안정성
    - 데이터 분석 솔루션 : 시스템을 세밀하기 조정하기 위해서는 데이터 수집과 분석이 중요