9️⃣

231011 9. 웹 크롤러 설계

9. 웹 크롤러 설계

웹 크롤러

검색 엔진에서 널리 쓰는 기술

웹에 새로 올라오거나 갱신된 콘텐츠(웹 페이지, 이미지, 비디오, pdf, ...)를 찾아내는 것이 주 목적

몇 개 웹 페이지에서 시작하여 그 링크를 따라 나가면서 새로운 콘텐츠를 수집

복잡도는 처리해야 하는 데이터의 규모에 따라 달라짐

크롤러 사용 예시

- 검색 엔진 인덱싱 (Search Engine Indexing)
  - 크롤러의 가장 보편적인 용례
  - 검색 엔진을 위한 로컬 인덱스를 만듦
  - google bot: 구글 검색엔진이 사용하는 웹 크롤러
- 웹 아카이빙
  - 나중에 사용할 목적으로 장기보관하기 위해 웹에서 정보를 모으는 절차
  - 많은 국립 도서관이 크롤러를 돌려 웹사이트를 아카이빙
- 웹 마이닝
  - 웹 마이닝을 통해 인터넷에서 유용한 지식을 도출해 낼 수 있음
- 웹 모니터링
  - 크롤러를 사용하면 인터넷에서 저작권이나 상표권이 침해되는 사례를 모니터링할 수 있음

▼ 1. 문제 이해 및 설계 범위 확정

웹 크롤러의 기본 알고리즘

1. 주어진 URL 집합의 URL들이 가리키는 모든 웹 페이지를 다운로드
2. 다운받은 웹 페이지에서 URL들을 추출
3. 추출된 URL들을 다운로드할 URL 목록에 추가하고 과정 반복

웹 크롤러가 만족시켜야 할 특성

- 규모 확장성
  - 병행성을 활용하면 보다 효과적으로 웹 크롤링을 할 수 있음
- 안정성
  - 비정상적 입력이나 환경에 대응할 수 있어야 함
  - 웹의 함정: 잘못 작성된 HTML, 아무 반응이 없는 서버, 장애, 악성코드가 붙어 있는 링크
- 예절
  - 수집 대상 웹 사이트에 짧은 시간 동안 너무 많은 요청을 보내서는 안됨
- 확장성
  - 새로운 형태의 콘텐츠를 지원하기가 쉬워야 함

개략적 규모 추정

- 매달 10억개의 웹 페이지를 다운로드
- QPS = 10억 / 30일 / 24시간 / 3600초 = 대략 400페이지/초
- 최대(peak) QPS = 2 * QPS = 800
- 웹 페이지의 크기 평균은 500k라고 가정
- 10억 페이지 * 500k = 500TB/월
- 1개월치 데이터를 보관하는 데는 500TB
- 5년간 보관한다고 가정할 경우 30PB(500TB * 12개월 * 5년)의 저장용량이 필요

▼ 2. 개략적 설계안 제시 및 동의 구하기

▼ 시작 URL 집합

웹 크롤러가 크롤링을 시작하는 출발점

전체 웹 크롤링이 필요한 경우 가능한 많은 링크를 탐색할 수 있도록 하는 URL을 고르는 것이 바람직

일반적으로는 전체 URL 공간을 작은 부분집합으로 나누는 전략을 사용

시작 URL을 무엇을 쓸 것이냐는 질문에 정답은 없으니까 의도가 무엇인지만 정확하게 전달하도록

▼ 미수집 URL 저장소

대부분의 웹 크롤러는 **다운로드할 URL**과 **다운로드된 URL** 두 가지로 나누어서 관리

다운로드할 URL = 미수집 URL 저장소

FIFO 큐라고 생각하면 됨

▼ HTML 다운로더

인터넷에서 웹 페이지를 다운로드하는 컴포넌트

다운로드할 페이지의 URL은 미수집 URL 저장소가 제공

▼ 도메인 이름 변환기

웹 페이지를 다운받으려면 URL을 IP 주소로 변환하는 절차가 필요

HTML 다운로더가 도메인 이름 변환기를 사용하여 IP 주소를 알아냄

▼ 콘텐츠 파서

웹 페이지를 다운로드하면 파싱과 검증 절차를 거쳐야 함

이상한 웹 페이지는 문제를 일으킬 수도 있고 저장 공간이 낭비됨

크롤링 서버 안에 파서를 구현하면 크롤링 과정이 느려질 수 있기 때문에 독립된 컴포넌트로 만드는 것을 추천

▼ 중복 콘텐츠인가?

웹에 공개된 연구결과에 따르면 29% 가량의 웹 페이지 컨텐츠는 중복

중복 컨텐츠 문제를 해결하기 위한 자료구조를 도입하여 중복을 줄이고 처리에 소요되는 시간을 줄이도록 함

두 HTML 파일을 효율적으로 비교하기 위해 해시 값 사용

▼ 콘텐츠 저장소

HTML 문서를 보관하는 시스템

저장할 데이터의 유형, 크기, 저장소 접근 빈도, 데이터의 유효 기간 등을 종합적으로 고려해서 구현 기술 선택

이번 설계안의 경우에는 **디스크와 메모리를 동시에 사용하는 저장소**를 택할 것

- 데이터 양이 너무 많기 때문에 대부분의 컨텐츠는 디스크에 저장
- 인기있는 컨텐츠는 메모리에 두어 접근 지연시간을 줄일 것

▼ URL 추출기

저장한 HTML 문서들을 파싱해 링크들을 골라내는 역할

상대경로는 전부 절대경로로 변환

▼ URL 필터

특정 조건의 URL을 배제함

- 특정한 컨텐츠 타입이나 파일 확장자를 갖는 URL
- 접속 시 오류가 발생하는 URL
- 접근 제외 목록(deny list) 에 포함된 URL

▼ 이미 방문한 URL?

이미 방문한 URL이나 미수집 URL 저장소에 보관된 URL을 추적할 수 있도록 하는 자료구조 사용

같은 URL을 여러 번 처리하는 일을 방지

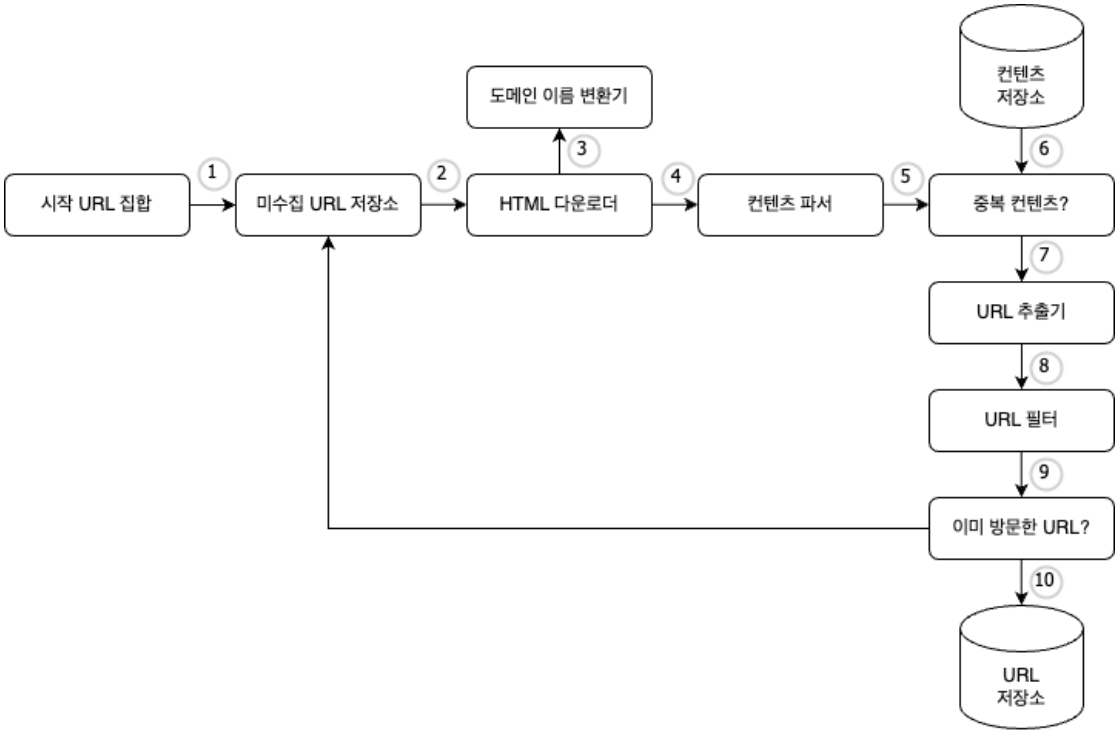
서버 부하를 줄이고 시스템이 무한 루프에 빠지는 일을 방지

블룸 필터나 **해시테이블**이 널리 쓰임

▼ URL 저장소

이미 방문한 URL을 보관하는 저장소

▼ 웹 크롤러 작업 흐름



1. 시작 URL들을 미수집 URL 저장소에 저장
2. HTML 다운로더는 미수집 URL 저장소에서 URL 목록을 가져옴
3. HTML 다운로더는 도메인 이름 변환기를 사용하여 URL의 IP 주소를 알아냄
4. 해당 IP 주소로 접속하여 웹 사이트 다운
5. 콘텐츠 파서는 다운된 HTML 페이지를 파싱하여 올바를 형식을 갖춘 페이지인지 검증
6. 콘텐츠 파싱과 검증이 끝나면 중복 콘텐츠인지 확인하는 절차를 개시
7. 중복 콘텐츠인지 확인하기 위해 해당 페이지가 이미 저장소에 있는지 확인
8. 이미 저장소에 있는 콘텐츠라면 처리하지 않고 버림
9. 없는 콘텐츠라면 저장소에 저장한 뒤 URL 추출기로 전달
10. URL 추출기는 해당 HTML 페이지에서 링크를 골라냄
11. 골라낸 링크를 URL 필터로 전달
12. 필터링이 끝나고 남은 URL만 중복 URL 판별 단계로 전달
13. 이미 처리한 URL인지 확인하기 위해 URL 저장소에 보관된 URL인지 살핌
14. 이미 저장소에 있는 URL인 경우 버림
15. 없는 URL인 경우 URL 저장소에 저장 후 미수집 URL 저장소에 전달

▼ 3. 상세 설계

▼ DFS를 쓸 것인가, BFS를 쓸 것인가

웹은 유향 그래프와 같음

페이지: 노드

하이퍼링크: 에지

깊이 우선 탐색법: DFS

좋은 선택이 아닐 가능성이 높음

그래프의 크기가 클 경우에는 어느 정도로 깊숙이 가게 될지 가늠하기 어렵기 때문에

너비 우선 탐색법: BFS

FIFO 큐를 사용하는 알고리즘

한쪽으로는 탐색할 URL를 집어넣고, 다른 한쪽으로는 꺼내기만 함

두 가지의 문제점

- 한 페이지에서 나오는 링크의 상당수는 같은 서버로 되돌아감
  - 같은 호스트에 속한 많은 링크를 다운받느라 바빠짐
  - 병렬로 처리하게 될 경우에는 해당 서버가 과부하에 걸릴 가능성이 높음
  - 이런 크롤러는 보통 '예의 없는' 크롤러라고 간주
- URL 간에 우선순위를 두지 않음
  - 처리 순서에 있어 모든 페이지를 공평하게 대우
  - 모든 웹 페이지가 같은 수준의 품질과 중요성을 갖지 않음
  - 페이지 순위, 사용자 트래픽의 양, 업데이트 빈도 등 여러 가지 척도를 통해 우선순위를 구별하는 것이 온당할 것

▼ 미수집 URL 저장소

미수집 URL 저장소를 활용하면 위의 문제를 좀 쉽게 해결할 수 있음

미수집 URL 저장소를 잘 구현하여 '예의'를 갖춘, 우선순위와 신선도를 구별하는 크롤러 구현 가능

예의

수집 대상 서버로 짧은 시간 안에 너무 많은 요청을 보내는 것을 삼가야 함

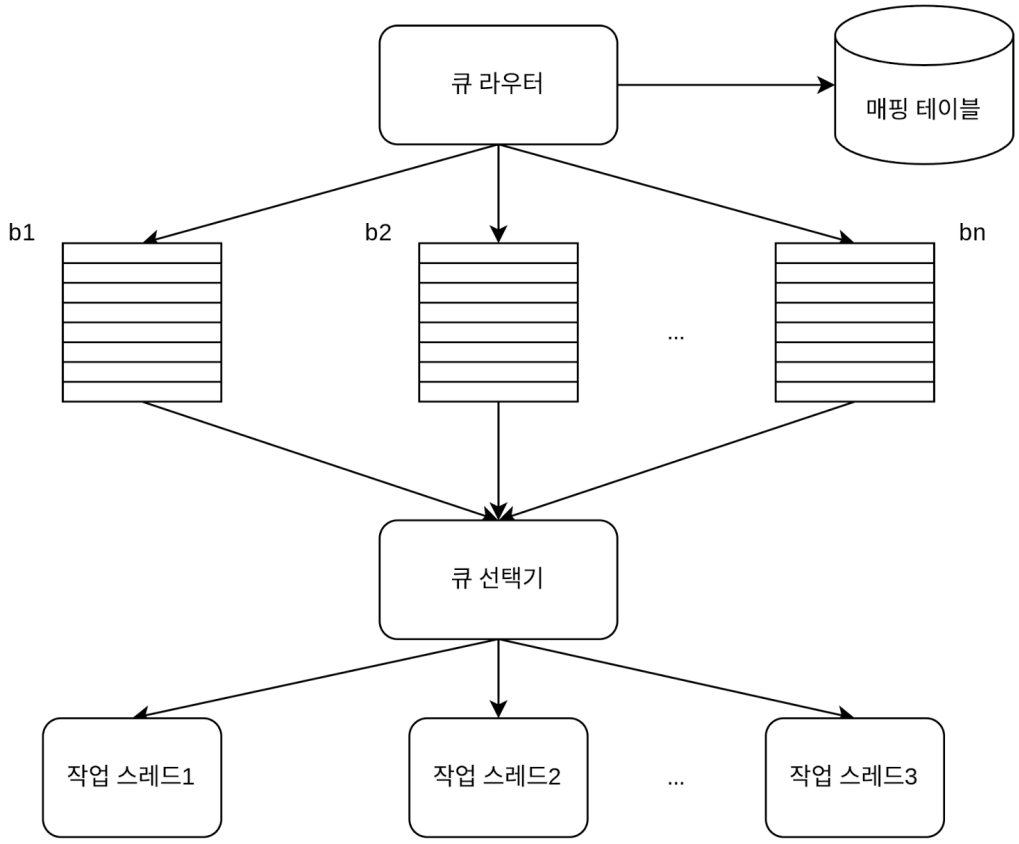
너무 많은 요청을 보내는 것은 '무례한' 일로 Dos 공격으로 간주되기도 함

동일 웹 사이트에 대해서는 한 번에 한 페이지만 요청

같은 웹 사이트의 페이지를 다운받은 태스크는 시간차를 두고 실행하도록

웹 사이트의 호스트명과 다운로드 수행하는 잡업 스레드 사이의 관계를 유지하면 됨

각 다운로드 스레드는 별도 FIFO 큐를 가지고 있어서 해당 큐에서 꺼낸 URL 만 다운로드



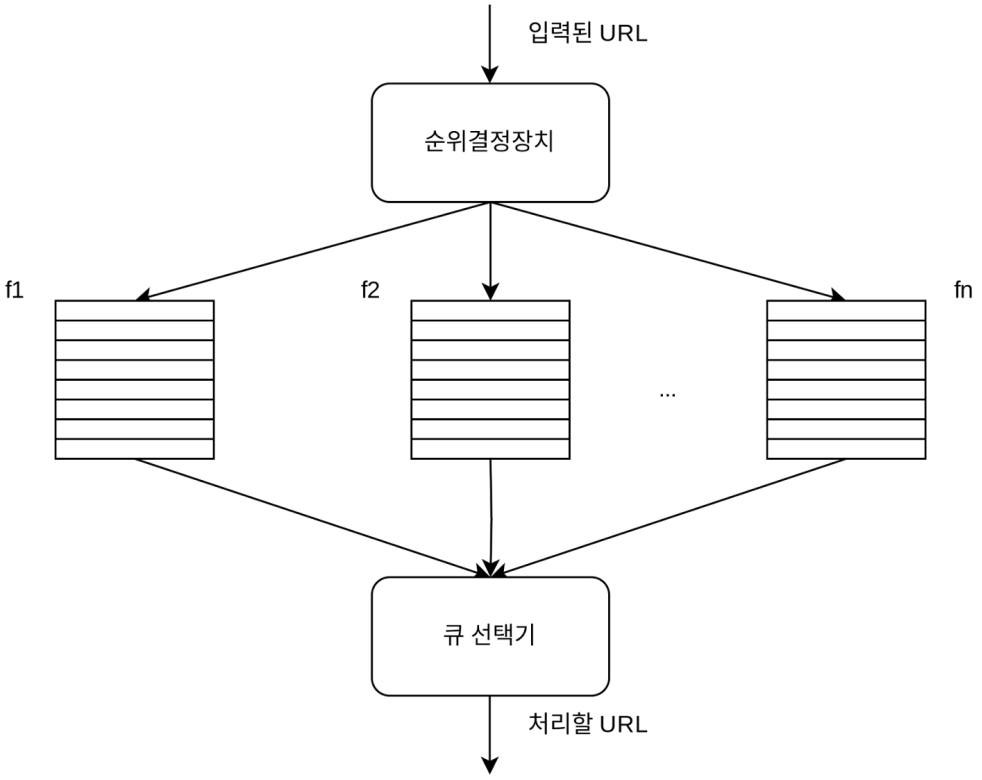
예의를 고려한 설계

- 큐 라우터
  - 같은 호스트에 속한 URL은 언제나 같은 큐로 가도록 보장하는 역할
- 매핑 테이블
  - 호스트 이름과 큐 사이의 관계를 보관하는 테이블
- FIFO 큐
  - 같은 호스트에 속한 URL은 언제나 같은 큐에 보관

- 큐 선택기
  - 큐 선택기는 큐들을 순회하면서 큐에서 URL을 꺼내서 해당 큐에서 나온 URL을 다운로드하도록 지정된 작업 스레드에 전달하는 역할
- 작업 스레드
  - 작업 스레드는 전달된 URL을 다운로드하는 작업을 수행
  - 전달된 URL은 순차적으로 처리
  - 작업들 사이에는 일정한 지연시간을 둘 수 있음

우선순위

유용성에 따라 우선순위를 나눌 때는 페이지 랭크, 트래픽 양, 갱신 빈도 등 다양한 척도를 사용할 수 있음

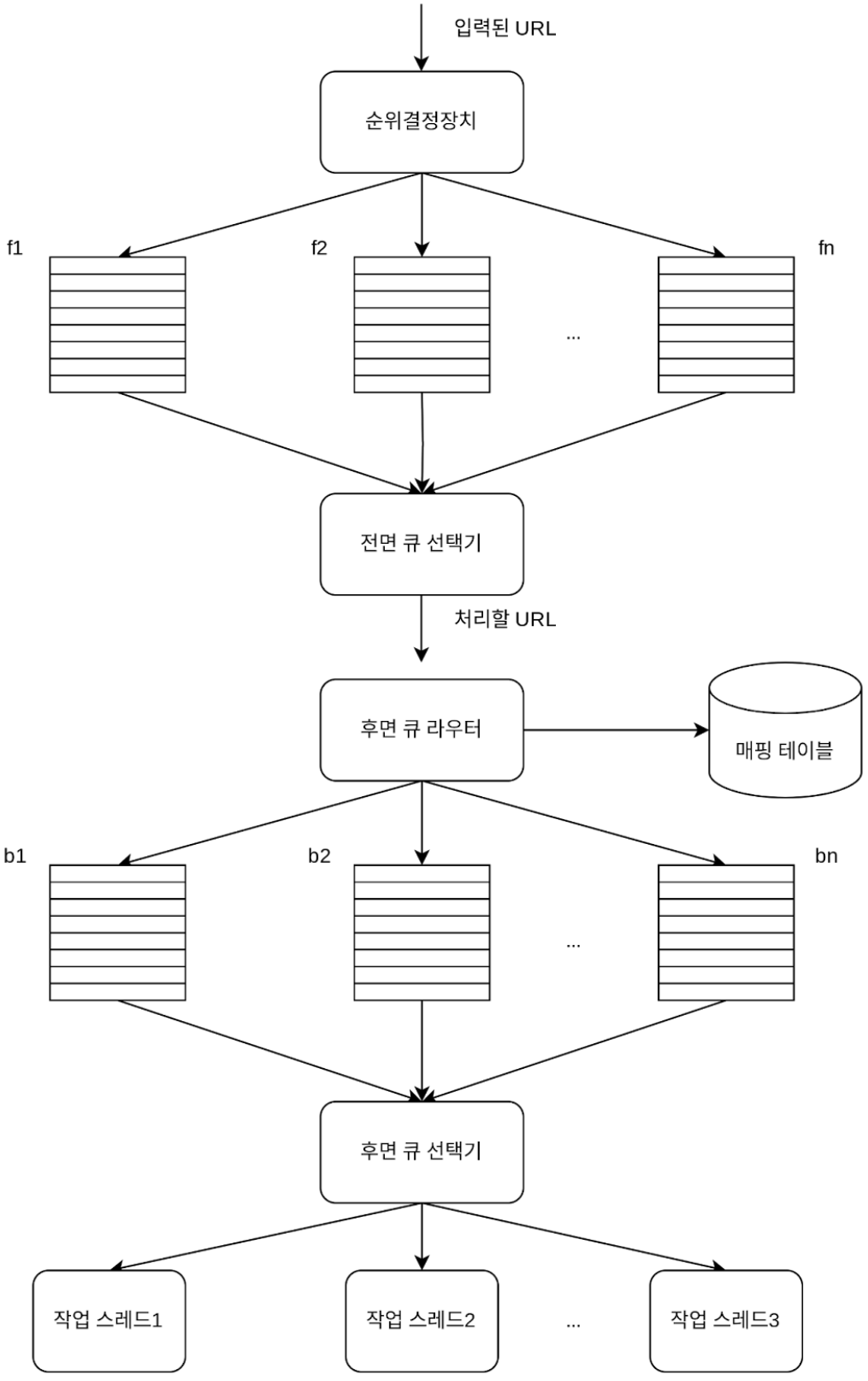


URL 우선순위를 고려한 설계

1. 순위 결정 장치(URL 우선순위를 정하는 컴포넌트)
   - URL 을 입력으로 받아 우선순위를 계산
2. 우선순위 별로 큐가 하나씩 할당

3. 우선순위가 높으면 선택될 확률도 높아짐
4. 큐 선택기
    - 임의 큐에서 처리할 URL을 꺼낸다. 순위가 높은 큐에서 더 자주 꺼내도록 프로그램 되어있다

예의 + 우선순위



여의와 우선순위를 모두 고려한 설계

- 전면 큐

    - 우선순위 결정 과정을 처리
- 후면 큐
    - 크롤러가 예의 바르게 동작하도록 보증

신선도

웹 페이지는 수시로 추가, 삭제, 변경됨

데이터의 신선함을 유지하기 위해서는 이미 다운로드한 페이지도 주기적으로 재수집할 필요가 있음

모든 URL을 재수집하는 것은 비효율적

재수집 작업을 최적화하기 위한 전략

- 웹 페이지의 변경 이력 활용
- 우선순위를 활용하여 중요한 페이지는 좀 더 자주 재수집

미수집 URL 저장소를 위한 지속성 저장장치

검색 엔진을 위한 크롤러는 처리해야 하는 URL의 수가 수억 개 이상

모든 것을 메모리에 보관하는 것은 안정성이나 규모 확장성 측면에서 바람직하지 않음

디스크에 저장하는 것도 느려서 쉽게 병목지점이 되는 이슈로 비추

⇒ 대부분의 URL은 디스크에 두지만 IO 비용을 줄이기 위해 메모리 버퍼에 큐를 두는 절충안 선택

⇒ 버퍼에 있는 데이터는 주기적으로 디스크에 기록

▼ HTML 다운로더

Robots.txt

로봇 제외 프로토콜

웹 사이트가 크롤러와 소통하는 표준

크롤러가 수집해도 되는 페이지 목록이 들어있음

웹 사이트를 다운받기 전에 해당 파일에 나열된 규칙을 먼저 확인해야 함

해당 파일을 중복으로 다운로드 하는 것을 피하기 위해 다시 다운 받아 캐시에 보관

성능 최적화

HTML 다운로더에 사용할 수 있는 성능 최적화 기법

- 분산 크롤링
  - 성능을 높이기 위해 크롤링 작업을 여러 서버에 분산
  - 각 서버는 여러 스레드를 돌려 다운로드 작업을 처리
  - URL 공간을 작은 단위로 분할
- 도메인 이름 변환 결과 캐시
  - 도메인 이름 변환기
    - 크롤러 성능의 병목 중 하나
    - DNS 요청을 보내고 결과를 받는 작업의 동기적 특성 때문
  - DNS 요청의 결과를 받기 전까지는 다음 작업을 진행할 수 없음
  - 크롤러 스레드 가운데 어느 하나라도 도메인 이름 변환기에 요청을 보내면 해당 요청이 완료될 때까지 다른 스레드의 요청이 모두 블록됨
  - DNS 조회 결과로 얻어진 도메인 이름과 그에 상응하는 IP 주소를 캐시에 보관하고 주기적으로 갱신하도록 구현하여 성능을 높일 수 있음
- 지역성
  - 크롤링 작업을 수행하는 서버를 지역별로 분산
  - 크롤링 서버가 크롤링 대상 서버와 지역적으로 가까우면 페이지 다운로드 시간이 줄어듦
  - 지역성을 활용하는 전략은 크롤 서버, 캐시, 큐, 저장소 등 대부분의 컴포넌트에 적용 가능
- 짧은 타임아웃
  - 어떤 웹서버는 응답이 느리거나 아예 응답하지 않음
  - 대기 시간이 길어지면 좋지 않기 때문에 크롤러가 최대 얼마나 기다릴지를 미리 정함
  - 타임아웃이 걸리면 해당 페이지 다운로드를 중단

안정성

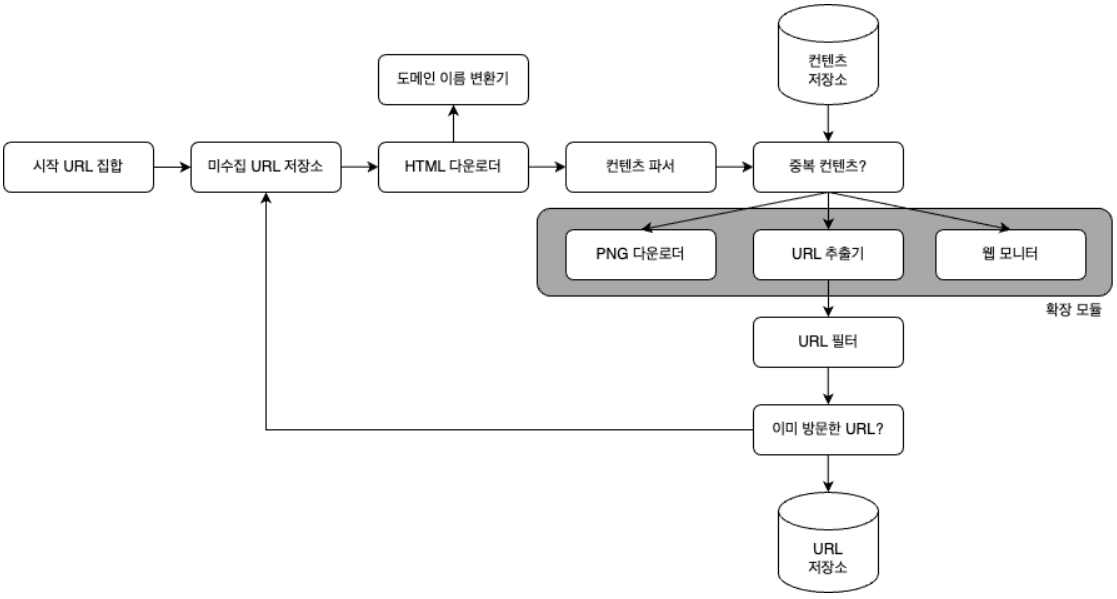
안정성은 성능 최적화 만큼 중요하게 고려해야할 부분

시스템 안정성을 향상시키기 위한 접근법

- 안정 해시
  - 다운로더 서버들에 부하를 고르게 분산하기 위해 적용 가능한 기술
  - 다운로더 서버를 쉽게 추가, 삭제 가능
- 크롤링 상태 및 수집 데이터 저장
  - 장애가 발생한 경우에도 쉽게 복구할 수 있도록 크롤링 상태와 수집된 데이터를 지속적 저장장치에 기록해두는 것이 바람직
  - 중단된 크롤링을 쉽게 재시작 가능
- 예외 처리
  - 예외가 발생해도 전체 시스템이 중단되는 일 없도록
- 데이터 검증
  - 시스템 오류를 방지하기 위한 중요 수단

확장성

새로운 형태의 컨텐츠를 쉽게 지원할 수 있도록 신경 써야 함



새로운 모듈을 끼워 넣음으로써 새로운 형태의 콘텐츠를 지원할 수 있도록 설계

- PNG 다운로더
  - PNG 파일을 다운로드 하는 plug-in 모듈
- 웹 모니터

    - 웹을 모니터링하여 저작권이나 상표권이 침해되는 일을 막는 모듈

문제 있는 콘텐츠 감지 및 회피

중복이거나 의미가 없거나 유해한 콘텐츠를 감지하고 시스템으로부터 차단하는 방법

1. 중복 컨텐츠
    - 웹 컨텐츠의 30% 가량은 중복
    - 해시, 체크섬을 사용하여 중복 콘텐츠 탐지
2. 거미 덫
    - 크롤러를 무한루프에 빠뜨리도록 설계한 웹 페이지
    - URL의 최대 길이를 제한
        - 모든 경우를 다 피할 수는 없음
    - 사람이 수작업으로 찾아낸 후 이런 사이트를 크롤러 탐색 대상에서 제외하거나 URL 필터 목록에 걸어둠
3. 데이터 노이즈
    - 어떤 컨텐츠는 거의 가치가 없음
    - 광고, 스크립트 코드, 스팸 URL
    - 이러한 컨텐츠를 가능한 제외해야 함

▼ 4. 마무리

추가로 논의해보면 좋은 이슈

- 서버 측 렌더링
    - 많은 웹사이트가 자바스크립트, ajax 등의 기술을 사용해서 링크를 즉석에서 만들어 냄
    - 웹 페이지를 그냥 있는 그대로 다운받아서 파싱해보면 그렇게 동적으로 생성되는 링크는 발견할 수 없음
    - 페이지를 파싱하기 전에 서버 측 렌더링을 적용하면 해결할 수 있음
- 원치 않는 페이지 필터링
    - 저장 공간 등 크롤링에 소요되는 자원은 유한
    - 스팸 방지 컴포넌트를 두어 품질이 조악하거나 스팸성인 페이지를 걸러내도록 해 두면 좋음

- 데이터베이스 다중화 및 샤딩
  - 데이터 계층의 가용성, 규모 확장성, 안정성 향상
- 수평적 규모 확장성
  - 대규모의 크롤링을 위해서는 다운로드를 실행할 서버가 수백 혹은 수천 대 필요하게 될 수도 있음
  - 무상태 서버
- 가용성, 일관성, 안정성
  - 성공적인 대형 시스템을 만들기 위해 필수적으로 고려해야 함
- 데이터 분석 솔루션
  - 시스템을 세밀히 조정하기 위해서 이런 데이터와 그 분석 결과가 필수적

▼ 토론

웹 크롤러를 개발하고 사용할 때 고려해야 할 윤리적 이슈와 어떻게 지켜야 할지(로봇 제외 프로토콜 방법 제외하고)

1. 개인 정보 보호:
   - 크롤링하면서 수집하는 데이터에는 개인 정보가 포함될 수 있습니다. 개인 식별 정보를 수집하려면 해당 사람들의 동의를 얻어야 합니다.
   - 크롤링된 데이터를 저장하고 사용할 때 개인 정보 보호 법규에 따라 안전하게 보호해야 합니다. 예를 들어, GDPR(일반 데이터 보호 규정) 또는 CCPA(캘리포니아 소비자 개인 정보 보호 법)와 같은 법규를 준수해야 합니다.
2. 저작권:
   - 크롤링을 통해 수집한 콘텐츠, 이미지 또는 기타 자산은 저작권 보호를 받을 수 있습니다. 크롤러를 사용하여 다른 사이트에서 콘텐츠를 가져올 때 저작권을 존중해야 합니다.
   - 사이트의 "로봇 배제 표준"을 존중하고, 크롤링이 허용되는지 확인해야 합니다. 로봇 배제 표준은 웹 사이트 운영자가 크롤러의 액세스를 허용 또는 제한하는 방법을 제공합니다.
3. 부하 및 대역폭:
   - 크롤러를 사용할 때 너무 빠르고 과도한 요청을 보내면 대상 웹 서버에 부하를 주고 서버 다운을 유발할 수 있습니다. 이로 인해 웹 사이트 운영자의 불편을 초래할 수 있으므로 적절한 요청 속도를 유지해야 합니다.

- 크롤링할 때 적절한 대역폭을 사용하여 대상 웹 서버의 리소스를 공정하게 분배해야 합니다.

4. 투명성:
    - 크롤러를 개발하고 사용할 때 그 목적을 명확하게 밝히고 투명해야 합니다. 데이터 수집의 목적과 사용 방법을 명시적으로 밝히는 것이 중요합니다.
    - 크롤링 대상 웹 사이트에 연락하여 크롤링 활동을 알리는 것도 고려해볼 만합니다.
5. 허용된 범위 내에서 작업:
    - 크롤링은 웹 사이트 운영자의 정책에 따라 금지될 수 있습니다. 크롤러를 개발하고 사용할 때는 대상 웹 사이트의 로봇 배제 표준을 준수하고, 웹 사이트 운영자의 정책을 따라야 합니다.