5주차

8장. URL 단축기 설계

1. 1단계 - 문제 이해 및 설계 범위 확정

면접장에서 시스템을 성공적으로 설계하기 위해서는 질문을 통해 모호함을 줄이고 요구사항을 알아내야 함

만들어야 하는 단축 URL의 개략적 추정>

- 쓰기 연산: 매일 1억개의 단축 URL 생성
- 초당 쓰기 연산: 1억(100million) / 24 / 3600 = 1160
- 읽기 연산: 읽기 연산과 쓰기 연산 비율은 10:1 → 읽기 연산은 초당 11,600회 발생
- URL 단축 서비스를 10년간 운영한다고 가정하면 1억 (100milion) x 365 × 10 = 3650억
- 축약 전 URL의 평균 길이는 100
- 10년 동안 필요한 저장 용량은 3650억 x 100 바이트 = 36.5TB

2. 2단계 - 개략적 설계안 제시 및 동의 구하기

API 엔드포인트

클라이언트는 서버가 제공하는 API 엔드포인트를 통해서 서버와 통신 → REST로 설계 진행

URL 단축기가 필요로 하는 엔드포인트

1. URL 단축용 엔드포인트: 새 단축 URL을 생성하고자 하는 클라이언트는 해당 엔드포인트에 단축할 URL을 인자로 실어서 POST 요청을 보냄

```
POST /api/v1/data/shorten
- 인자: {longURL: longURLstring}
- 반환: 단축 URL
```

2. URL 리디렉션용 엔드포인트: 단축 URL에 대해서 HTTP 요청이 오면 원래 URL로 보내주기 위한 용도의 엔드포인트

```
GET /api/v1/shortUrl
- 반환: HTTP 리디렉션 목적지가 될 원래 URL
```

브라우저에 단축 URL을 입력하면 단축 URL 을 받은 서버는 그 URL을 원래 URL로 바꾸어 301응답의 location헤더에 넣어 반환을 진행

- 301 Premanently Moved: 해당 URL에 대한 http 요청의 처리 책임이 영구적으로 location 헤더에 반환된 url로 이전되었다는 응답
  - 서버 부하를 줄이는 것이 중요할 때 사용
- 302 Found: 주어진 URL로의 요청이 일시적으로 lcoation 헤더가 지정하는 url에 의해 처리되어야 한다는 응답
  - 트래픽 분석이 중요할 때 유리

URL 리디렉션 구현 방법

- 해시 테이블을 사용하기 → 가장 직관적
  - 해시 테이블에 <단축 URL , 원래 URL>의 쌍을 저장할 경우
    - 원래 URL = hashTable.get(단축 URL)
    - 301혹은 302 응답 location 헤더에 원래 url을 넣은 후 전송

URL 단축

단축 URL을 만드는 해시 함수의 요구사항

- 입력으로 주어지는 긴 URL이 다른 값이면 해시 값도 달라야 한다.

- 계산된 해시 값은 원래 입력으로 주어졌던 긴 URL로 복원될 수 있어야 한다.

3. 3단계 - 상세 설계

데이터 모델

메모리는 유한하고 비싸기 때문에 해시 테이블에 모든 걸 올려두는 방법은 실제 시스템에 쓰기 어려움 → <단축 URL, 원래 URL>의 순서쌍을 관계형 데이터베이스에 저장하는 방법을 사용

해시 함수

hashValue: 해시 함수가 계산하는 단축 URL 값

해시 값 길이

- hashValue는 [0-9, a-z, A-Z]의 62개 문자를 사용할 수 있음
- 62^n ≥ 3650억인 n의 최솟값을 찾아야 함

n= 7일 때, 3.5조개의 URL을 만들 수 있으므로 hashValue의 길이는 7

해시 함수 구현에 쓰이는 기술

- 해시 후 충돌 해소
- base-62 변환

해시 후 충돌 해소

CRC32, MD5, SHA-1 등 잘 알려진 해시 함수를 사용하면 쉽게 축약 결과를 얻을 수 있지만, 원하는 길이인 7보다 모두 김

- 계산된 해시 값에서 처음 7개의 글자만 이용하기

- 서로 충돌할 확률이 높아짐 → 충돌이 해소될 때 까지 사전에 정의한 문자열을 해시 값에 덧붙이기
- 충돌은 해소할 수 있으나 단축 URL 생성시 한 번 데이터베이스를 질의 → 오버헤드 ↑
  - 데이터베이스 대신 블룸 필터를 사용하면 성능을 높일 수 있음
  - 블룸 필터: 어떤 집합에 특정 원소가 있는지 검사할 수 있도록 하는, 확률론에 기초한 공간 효율이 좋은 기술

base-62 변환

수의 표현 방식이 다른 두 시스템이 같은 수를 공유하여야 하는 경우에 유용

62진법을 쓰는 이유는 hashvalue에 사용할 수 있는 문자 개수가 62개이기 때문임

- 62진법은 수를 표현하기 위해 총 62개의 문자를 사용하는 진법 → 0은 0으로, 9는 9로, 10은 a로, 11은 b로, ..., 35는 A로, ... 61은 Z로 대응시켜 표현할 것
- $11157_{10} = 2 \times 62^2 + 55 \times 62^1 + 59 \times 62^0 = [2, 55, 59] \Rightarrow [2, T, X] \Rightarrow 2TX_{62}$

URL 단축기 상세 설계

URL 단축기는 시스템의 핵심 컴포넌트이므로, 그 처리 흐름이 논리적으로는 단순해야 하고 기능적으로는 언제나 동작하는 상태로 유지되어야 함

처리 흐름

1. 입력으로 긴 URL을 받는다.
2. 데이터베이스에 해당 URL이 있는지 검사한다.
3. 데이터베이스에 있다면 해당 URL에 대한 단축 URL을 만든 적이 있는 것이다. 따라서 데이터베이스에서 해당 단축 URL을 가져와서 클라이언트에게 반환
4. 데이터베이스에 없는 경우에는 해당 URL은 새로 접수된 것이므로 유일한 ID를 생성한다. 이 ID는 데이터베이스의 기본 키로 사용된다.
5. 62진법 변환을 적용, ID를 단축 URL로 만든다.
6. ID, 단축 URL, 원래 URL로 새 데이터베이스 레코드를 만든 후 단축 URL을 클라이언트에 전달

- ID 생성기: 단축 URL을 만들 때 사용하는 ID를 만드는 생성기, 전역적 유일성이 보장되어야 함

URL 리디렉션 상세 설계

1. 사용자가 단축 URL을 클릭한다.
2. 로드밸런서가 해당 클릭으로 발생한 요청을 웹 서버에 전달한다.
3. 단축 URL이 이미 캐시에 있는 경우에 원래 URL을 바로 꺼내서 클라이언트에게 전달한다.
4. 캐시에 해당 단축 URL이 없는 경우에는 데이터베이스에서 꺼낸다. 데이터베이스에 없다면 아마 사용자가 잘못된 단축 URL을 입력한 경우일 것이다.
5. 데이터베이스에서 꺼낸 URL을 캐시에 넣은 후 사용자에게 반환한다.

4. 4단계 - 마무리

시간이 남을 경우 추가적으로 하면 좋은 이야기

- 처리율 제한 장치
- 웹 서버의 규모 확장
- 데이터베이스의 규모 확장
- 데이터 분석 솔루션
- 가용성, 데이터 일관성, 안전성

9장. 웹 크롤러 설계

웹 크롤러: 검색 엔진에서 사용되는 기술. 웹에 새로 올라오거나 갱신된 콘텐츠를 찾아내는 것이 주된 목적. 콘텐츠는 웹 페이지, 이미지, 비디오 등을 다 포함

몇 개의 웹 페이지에서 시작하여 링크를 따라 나감 → 새로운 콘텐츠를 수집

웹 크롤러 이용 종류

- 검색 엔진 인덱싱
- 웹 아카이빙
- 웹 마이닝
- 웹 모니터링

웹 크롤러의 복잡도는 웹 크롤러가 처리해야 하는 데이터의 규모에 따라 달라짐

1. 1단계 - 문제 이해 및 설계 범위 확정

웹 크롤러의 기본 알고리즘

1. URL 집합이 입력으로 주어지면, 해당 URL들이 가리키는 모든 웹 페이지를 다운로드한다.
2. 다운받은 웹 페이지에서 URL들을 추출한다.
3. 추출된 URL들을 다운로드할 URL 목록에 추가하고 위의 과정을 처음부터 반복한다.

그러나 실제 웹 크롤러는 단순하지 않음 → 요구사항을 알아내고 설계 범위를 좁혀야 함

웹 크롤러가 만족해야 하는 속성

- 규모 확장성
- 안전성
- 예절: 크롤러는 수집 대상 웹 사이트에 짧은 시간 동안 너무 많은 요청을 보내서는 안됨
- 확장성

개략적 규모 추정

- 매달 10억개의 웹 페이지를 다운로드한다.

- QPS = 10억(1billion, 즉 1,000, 000, 000) / 30일 / 24시간 / 3600초 = 대략 400 페이지/초
- 최대 QPS = 2 x QPS = 800
- 웹 페이지의 크기 평균은 500k라고 가정
- 10억 페이지 x 500k = 500TB/월.

2. 2단계 - 개략적 설계안 제시 및 동의 구하기

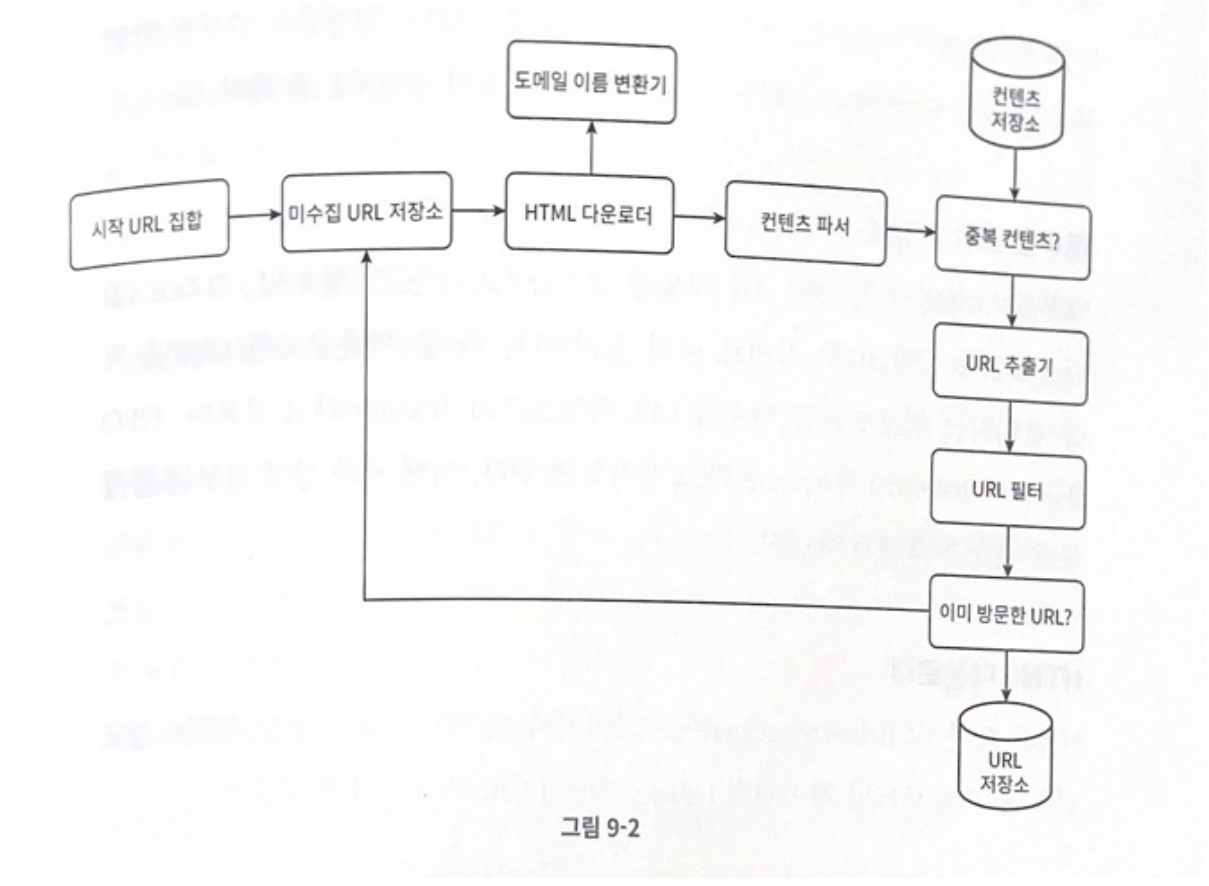

그림 9-2

시작 URL 집합

시작 URL 집합은 웹 크롤러가 크로링을 시작하는 출발점이 됨

전체 웹을 크롤링해야 하는 경우에는 시작 URL을 고를 때 좀 더 창의적일 필요가 존재 → 크롤러가 가능한 많은 링크를 탐색할 수 있도록 하는 URL을 고르는 것이 바람직함

시작 URL 집합 예시

- 나라별로 인기 있는 웹 사이트가 다르다는 점에 착안, 전체 URL 공간을 작은 부분집합으로 나누는 방법
- 주제별로 다른 시작 URL을 사용

미수집 URL 저장소

대부분 현대적 웹 크로러는 크롤링 상태를 다음과 같이 나누어 관리함

1. 다운로드할 URL
2. 다운로드된 URL

이 중, 다운로드 할 URL을 저장 관리하는 컴포넌트를 미수집 URL 저장소라고 부름 → FIFO 큐

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 다운로더는 도메인 이름 변환기를 사용하여 URL에 대응되는 IP 주소를 알아냄

콘텐츠 파서

웹 페이지를 다운로드하면 파싱과 검증 절차를 거쳐야 함 → 이상한 웹 페이지는 문제를 야기할 수 있음 + 저장 공간만 낭비하게 됨

크롤링 서버 안에 파서 구현 시 크롤링 과정이 느려질 수 있으므로 독립된 컴포넌트로 만들기

중복 콘텐츠인가?

웹 페이지의 해시 값을 비교하여 데이터 중복을 줄이고 데이터 처리에 소요되는 시간 줄이기

콘텐츠 저장소

HTML 문서를 보관하는 시스템

저장소 구현 시, 데이터의 유형, 크기, 저장소 접근 빈도, 데이터의 유효 기간 등을 종합적으로 고려해야 함

책에서 사용하는 콘텐츠 저장소

- 데이터 양이 너무 많으므로 대부분의 콘텐츠는 디스크에 저장한다.
- 인기 있는 콘텐츠는 메모리에 두어 접근 지연 시간을 줄일 것이다.

URL 추출기

HTML 페이지를 파싱하여 링크들을 골라내는 역할.

URL 필터

특정한 콘텐츠 타입이나 파일 확장자를 갖는 URL, 접속 시 오류가 발생하는 URL, 접근 제외 목록에 포함된 URL 등을 크롤링 대상에서 배제하는 역할

이미 방문한 URL?

이미 방문한 적이 있는 URL인지 추적하면 같은 URL을 여러 번 처리하는 일을 방지 할 수 있으므로 서버 부하를 줄이고시스템이 무한 루프에 빠지는 일을 방지할 수 있음

→ 해시 테이블, 블룸 필터 사용

URL 저장소

이미 방문한 URL을 보관하는 저장소

웹 크롤러 작업 흐름

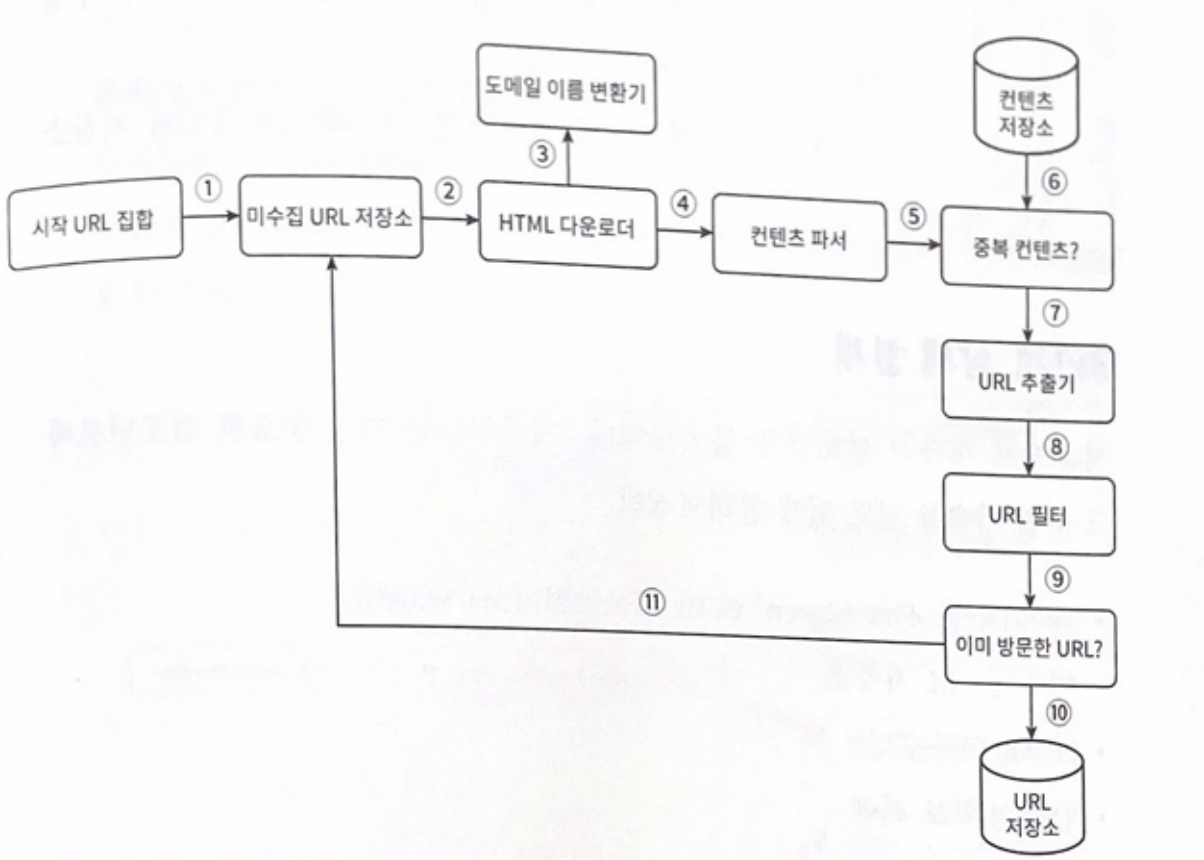

그림 9-4

3. 3단계 - 상세 설계

DFS vs BFS

웹은 유향 그래프와 같음

- 페이지는 노드, 하이퍼링크는 에지로 볼 수 있음
- 크롤링 프로세스는 유향 그래프를 에지를 따라 탐색하는 과정

웹 크롤러는 DFS와 BFS 중 보통 BFS, 즉 너비 우선 탐색법을 사용함 → BFS는 FIFO 큐를 사용

BFS 구현법의 문제점

- 예의 없는 크롤러 문제

    - 한 페이지에서 나오는 링크의 상당수는 같은 서버로 되돌아 감
        - 크롤러는 같은 호스트에 속한 많은 링크를 다운 받느라 바빠지는데, 이때 링크를 병렬로 처리하면 위키피디아 서버는 수 많은 요청으로 과부하에 걸리게 됨

      ⇒ 크롤러가 예의 없는 크롤러로 간주 됨
- 우선순위 문제
    - 표준적 BFS 알고리즘은 URL 간에 우선순위를 두지 않지만, 모든 웹 페이지가 같은 수준의 품질, 같은 수준의 중요성을 갖지 않음
        - 페이지 순위, 사용자 트래픽 양, 업데이트 빈도 등 여러 척도에 비추어 처리 우선순위를 구별해야 함

미수집 URL 저장소

미수집 URL 저장소를 잘 구현하면 “예의”를 갖춘 크롤러, URL 사이의 우선순위와 신선도를 구별하는 크롤러 구현이 가능함

예의

예의 바른 크롤러를 만들기 위해서는 동일 웹 사이트에 대해서는 한 번에 한 페이지만 요청해야 함

- 웹 사이트의 호스트명과 다운로드를 수행하는 작업 스레드 사이의 관계를 유지하여 해당 요구사항을 만족 시킬 수 있음
    - 다운로드 스레드가 별도로 FIFO 큐를 가지고, 해당 큐에서 꺼낸 URL만 다운로드하게 함

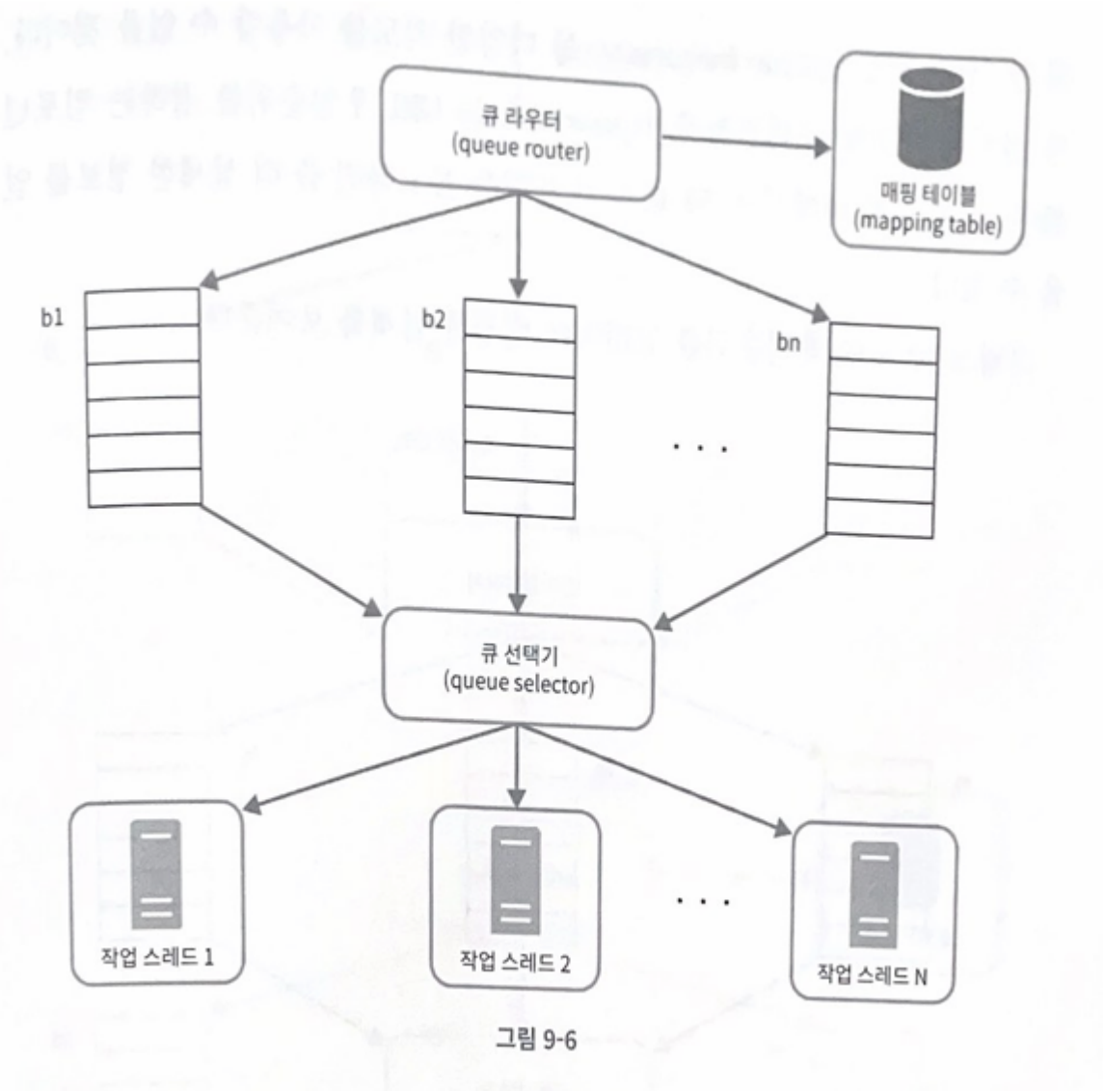

그림 9-6

- 큐 라우터: 같은 호스트에 속한 URL은 언제나 같은 큐로 가도록 보장하는 역할을 함
- 매핑 테이블: 호스트 이름과 큐 사이의 관계를 보관하는 테이블
- FIFO 큐: 같은 호스트에 속한 URL은 언제나 같은 큐에 보관됨
- 큐 선택기: 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 되며, 작업들 사이에는 일정한 지연시간을 둘 수 있음

우선순위

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

⇒ 순위 결정 장치를 사용하여 URL 우선순위를 결정함

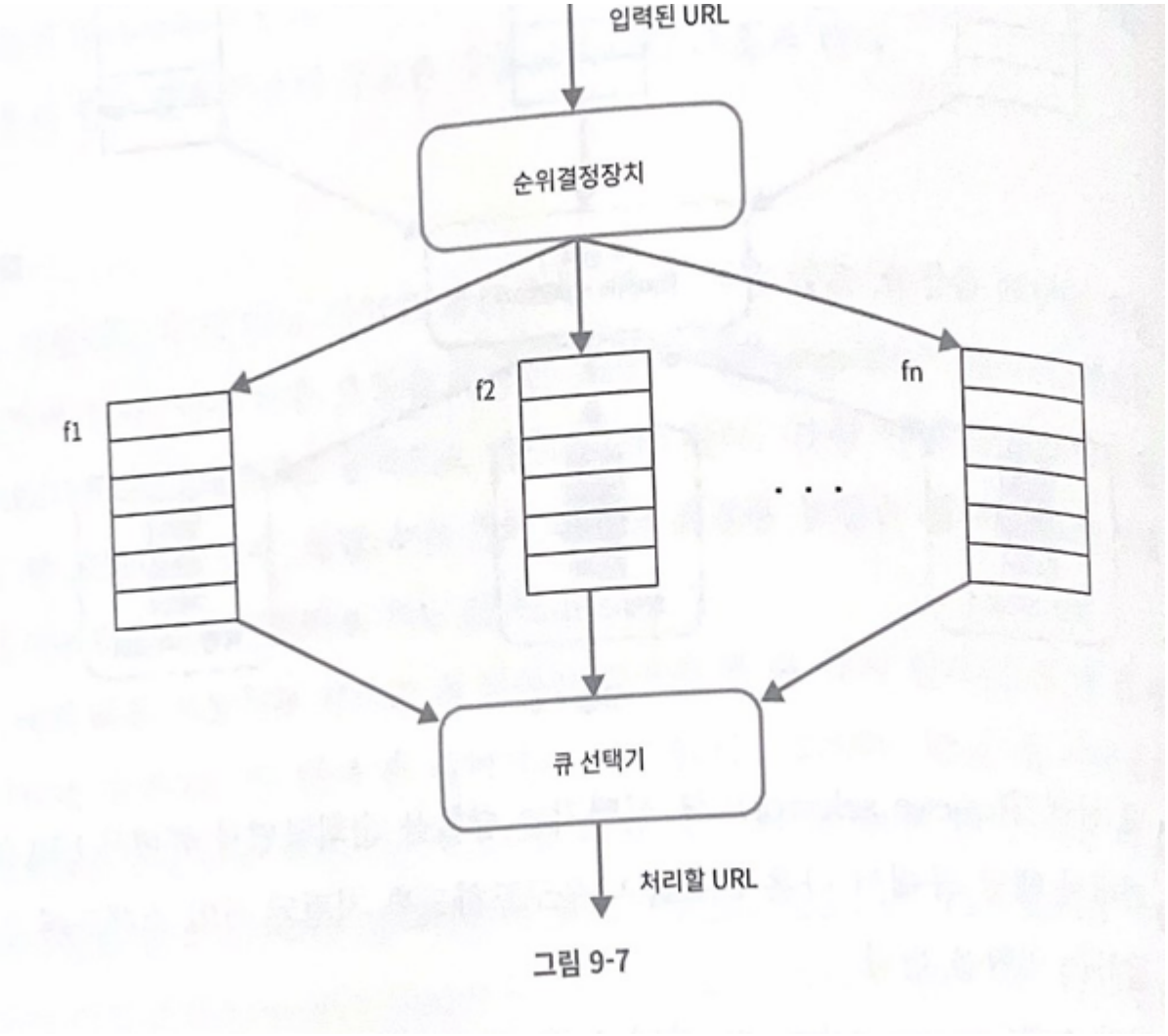

그림 9-7

- 순위 결정장치: URL을 입력으로 받아 우선순위를 계산
- 큐: 우선순위별로 큐가 하나씩 할당됨. 우선순위가 높으면 선택될 확률이 올라감
- 큐 선택기: 임의 큐에서 처리할 URL을 꺼내는 역할을 담당, 순위가 높은 큐에서 더 자주 꺼내도록 프로그램되어 있음

전체 설계

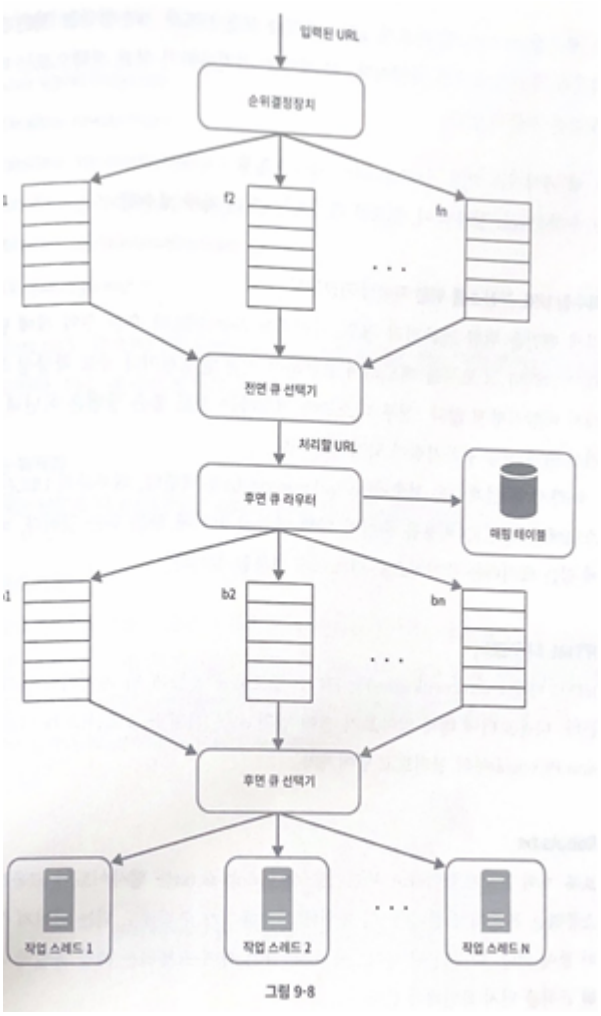

그림 9-8

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

신선도

웹 페이지는 수시로 추가되고 삭제되고 변경되므로 데이터의 신선함을 유지하기 위해서는 이미 다운로드한 페이지도 주기적으로 재수집해야 함

데이터 신선함 유지 작업을 최적화하는 전략

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

미수집 URL 저장소를 위한 지속성 저장장치

검색 엔진을 위한 크롤러의 경우 처리해야 하는 URL의 수가 많기 때문에 모드를 메모리에 보관하는 것은 안정성이나 규모 확장 측면에서 좋지 않음

그러나 전부 디스크에 저장하는 것도 느려서 쉽게 성능 병목지점이 되기 때문에 좋은 방법은 아님

→ 책에서는 절충안을 선택, 대부분의 URL은 디스크에 두지만 I/O 비용을 줄이기 위해 메모리 버퍼에 큐를 두어 버퍼에 있는 데이터를 주기적으로 디스크에 기록

HTML 다운로더

로봇 제외 프로토콜(Robots.txt)

웹 페이지가 크롤러와 소통하는 표준적 방법, 크롤러가 수집해도 되는 페이지 목록이 들어가 있음 → 웹 사이트를 긁어가기 전에 크롤러는 해당 파일에 나열된 규칙을 확인해야 함

성능 최적화

1. 분산 크롤링

성능을 높이기 위해 크롤링 작업을 여러 서버에 분산하는 방법. 각 서버는 여러 스레드를 돌려 다운로드 작업을 처리함

- 해당 구성을 위해 URL공간은 작은 단위로 분할하여, 각 서버는 그중 일부의 다운로드를 담당

2. 도메인 이름 변환 결과 캐시

도메인 이름 변환기는 DNS 요청을 보내고 결과를 받는 작업의 동기적 특성 때문에 크롤러 성능의 병목 중 하나가 됨.

따라서 DNS 조회 결과로 얻어진 도메인 이름과 IP 주소 사이의 관계를 캐시에 보관하고 크론 잡등을 돌려 주기적으로 갱신하도록 함 → 성능 ↑

3. 지역성

크롤링 작업을 수행하는 서버를 지역별로 분산하는 방법. 크롤링 서버와 지역적으로 줄어들게 됨

4. 짧은 타임아웃

최대 얼마나 기다릴지를 미리 정해두고, 이 시간 동안 서버가 응답하지 않으면 해당 페이지 다운로드를 중단하고 다음 페이지로 넘어가게 함

안전성

시스템 안전성을 향상시키기 위한 접근법 가운데 중요한 몇가지는 다음과 같음

- 안정해시: 다운로더 서버들에 부하를 분산할 때 적용 가능한 기술, 해당 기술을 이용하면 다운로더 서버를 쉽게 추가하고 삭제할 수 있음
- 크롤링 상태 및 수집 데이터 저장: 장애가 발생한 경우에도 쉽게 복구할 수 있도록 크롤링 상태와 수집된 데이터를 지속적 저장장치에 기록해두어야 함
- 예외 처리: 예외가 발생해도 전체 시스템이 중단되는 일 없이 그 작업을 우아하게 이어나갈 수 있도록 해야 함
- 데이터 검증: 시스템 오류를 방지하기 위한 중요 수단 중 하나

확장성

시스템을 설계할 때 새로운 형태의 콘텐츠를 쉽게 지원할 수 있도록 신경 써야 함

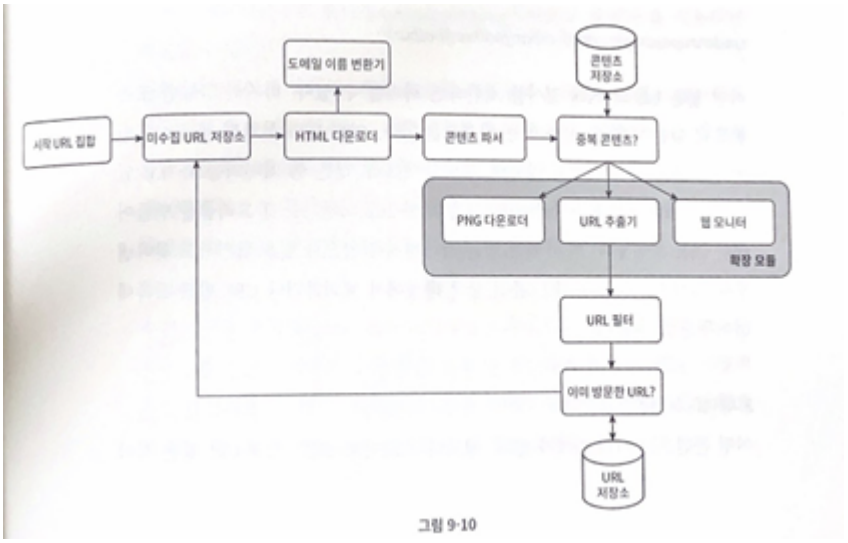

그림 9-10

- PNG 다운로더: PNG 파일을 다운로드하는 플러그인 모듈
- 웹 모니터: 웹을 모니터링하여 저작권이나 상표권이 침해되는 일을 막는 모듈

문제 있는 콘텐츠 감지 및 회피

1. 중복 콘텐츠

웹 콘텐츠의 30%는 중복 → 해시나 체크섬을 사용하면 중복 콘텐츠를 보다 쉽게 탐지 가능

2. 거미 덫

거미 덫: 크롤러를 무한 루프에 빠뜨리도록 설계한 웹 페이지

이러한 덫이 설치된 웹 사이트는 일반적으로 기이할 정도록 많은 웹 페이지를 가지고 있기 때문에 알아내기 쉬움

단, 덫을 자동으로 피해가는 알고리즘을 만들어내는 것은 까다로움

ex) 사람이 수작업으로 덫을 확인하고 찾아낸 후, 사이트를 크롤러 탐색 대상에서 제외하거나 URL 필터 목록에 걸어두는 방법

3. 데이터 노이즈

광고나 스크립트 코드, 스팸 URL 등은 크롤러에게 도움이 되지 않으므로 가능하다면 제외함

4. 4단계 - 마무리

시간이 남는다면 추가로 논의해볼만한 사항

- 서버 측 렌더링
- 원치 않는 페이지 필터링
- 수평적 규모 확장성
- 데이터베이스 다중화 및 샤딩
- 가용성, 일관성, 안전성
- 데이터 분석 솔루션