9장 - 웹 크롤러 설계

Tags

개요

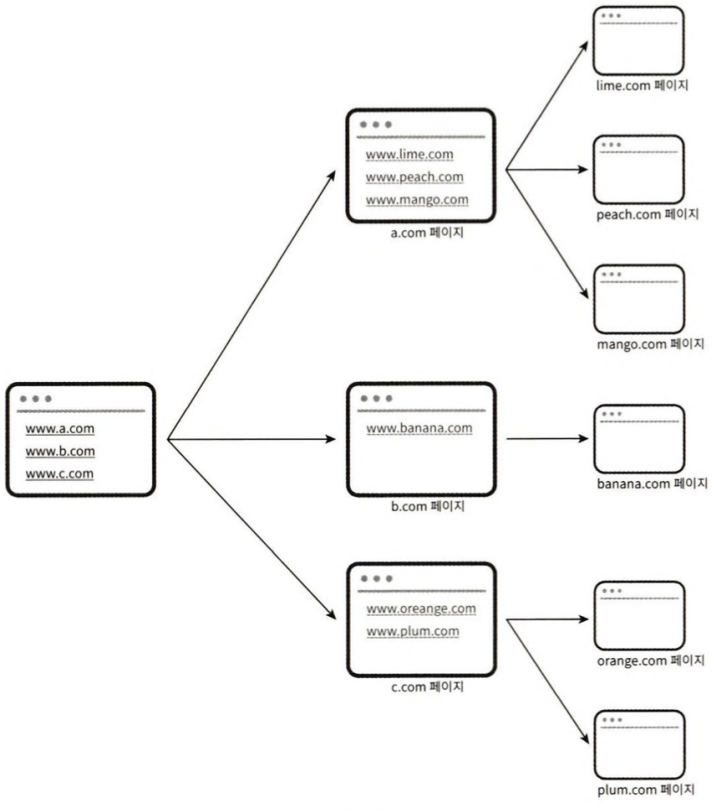
- 웹 크롤러는 로봇(robot) 또는 스파이더(spider)라고도 부른다.

  → 검색 엔진에 널리 쓰이는 기술, 웹 컨텐츠를 찾아내는 것이 주된 목적

- **검색 엔진 인덱싱(search engine indexing)**: 크롤러의 가장 보편적인 용례.

  → 크롤러는 웹 페이지를 모아 검색 엔진을 위한 로컬 인덱스(local index)를 만든다.

- **웹 아카이빙(web archiving)**: 나중에 사용할 목적으로 장기보관하기 위해 웹에서 정보를 모으는 절차를 말한다. 많은 국립 도서관이 크롤러를 만들어서 웹사이트를 아카이빙함.



- **웹 마이닝(web mining)**: 웹의 폭팔적 성장세는 데이터 마이닝(data mining)업계에 전례 없는 기회다. 인터넷에서 유용한 지식을 도출해 낼 수 있다. 유명 금융 기업들은 크롤러를 사용해 주주총회 자료나 연차 보고서(annual report)를 다운받아 기업의 핵심 사업을 정하기도 한다
- **웹 모니터링(web monitoring)**: 크롤러를 사용하면 인터넷 저작권이나 상표권이 침해 사례를 모니터링 할 수 있다.

문제 이해 및 설계 범위 확장

- 웹 크롤러의 기본 알고리즘
    1. URL 집합이 입력으로 주어지면, 해당 URL들이 가리키는 모든 웹 페이지를 다운로드 한다.
    2. 다운받은 웹 페이지에서 URL들을 추출한다.
    3. 추출된 URL들을 다운로드할 URL 목록에 추가하고 위의 과정을 처음부터 반복한다.

웹 크롤러의 속성

- 규모 확장성: 웹은 거대하다. 오늘날 웹에는 수십억 개의 페이지가 존재한다. 따라서 병행성(parallelism)을 활용 한다면 보다 효과적으로 이다.
- 안정성(robustness): 웹은 함정으로 가득하다. 잘못 작성된 HTML, 아무 반응 없는 서버, 장애, 악성 코드가 붙은 링크 등이다. 비정상적 입력이나 환경에 잘 대응해야 한다.
- 예절(politeness): 크롤러는 수집 대상 웹 사이트에 짧은 시간 동안 너무 많은 요청을 보내서는 안된다.
- 확장성(extensibility): 새로운 형태의 콘텐츠를 지원하기가 쉬워야 한다. 이미지 파일도 크롤링을 한다고 가정한다면 전체 시스템을 새로 설계해야 한다면 곤란할 것이다.

개략적 규모 추정

- 매달 10억 개의 웹페이지를 다운로드 한다.
- QPS = 10억(1bilion, 즉 1,000,000,000) / 30일 / 24시간 / 3600초 = 대량 400페이지/초

- 최대(Peak) Qps = 2 * QPS = 800
- 웹 페이지의 크기 평균 500K라고 가정
- 10억 페이지 * 500K = 500TB/월
- 1개월치 데이터를 보관하는 데는 500TB, 5년간 보관한다고 가정한다면 결국 500TB * 12개월 * 5년 = 30PB의 저장 룡양이 필요

개략적 설계안 제시

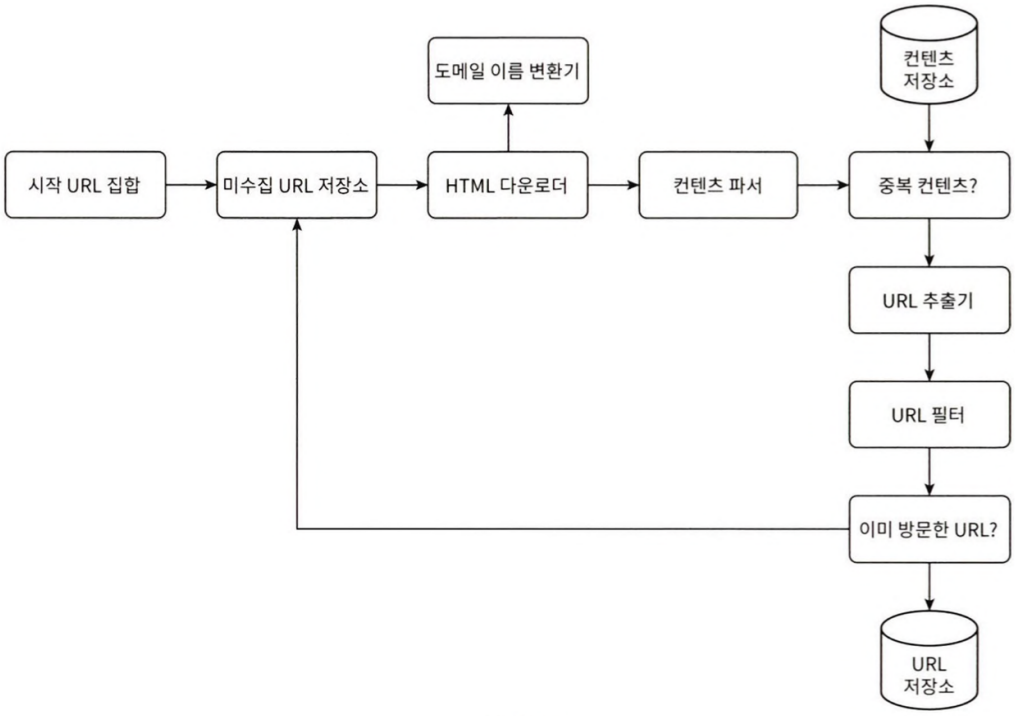


그림 9-2

시작 URL 집합

- 시작 URL 집합은 웹 크롤러가 크롤링을 시작하는 출발점이다.
    - 어떤 대학 웹사이트로부터 찾아 나갈 수 있는 모든 웹 페이지를 크롤링하는 가장 직관적인 방법은 해당 대학의 도메인 이름이 붙은 모든 페이지의 URL을 시작 URL로 쓰는 것이다.

- 전체 웹을 크롤링 해야하는 경우 시작 URL을 고를 때 좀 더 창의적일 필요가 있다.
    - 크롤러가 가능한 한 많은 링크를 탐색할 수 있도록 하는 URL을 고르는 것이 바람직하다.
    - 일반적으로는 전체 URL 공간을 작은 부분집합으로 나누는 전략을 쓴다.
    - 또 다른 방법은 주제별로 다른 시작 URL을 사용하는 것이다.
        - URL 공간을 쇼핑, 스포츠, 건강 등등 주제별로 세분화
- 시작 URL로 무엇을 쓸 것인지 정답은 없다.

미수집 URL 저장소

- 현대적 웹 크롤러는 크롤링 상태를 두 가지로 나누어 관리
    1. 다운로드 할 URL
    2. 다운로드된 URL
- 다운로드할 URL을 저장 관리하는 컴포넌트를 미수집 URL 저장소(URL fronier)라고 부른다.

  → FIFO(First-In-First-Out), 큐(Queue)라고 생각하면 된다.

HTML 다운로더

- HTML 다운로더(downloader)는 인터넷에서 웹 페이지를 다운로드하는 컴포넌트다.
- 다운로드할 페이지의 URL은 미수집 URL 저장소가 제공한다.

도메인 이름 변환기

- 웹 페이지를 다운받으려면 URL을 IP 주소로 변환하는 절차가 필요하다.
- HTML 다운로더는 도메인 이름 변환기를 사용하여 URL에 대응되는 IP 주소를 알아낸다.

콘텐츠 파서

- 웹 페이지를 다운로드하면 파싱(parsing)과 검증(validation) 절차를 거쳐야한다.
- 이상한 웹 페이지는 문제를 일으킬 수 있는데다 저장 공간만 낭비하게 된다.
- 크롤링 서버안에 콘텐츠 파서를 구현하면 크롤링 과정이 느려지게 될 수 있으므로, 독립된 컴포넌트로 만든다.

중복된 컨텐츠인가?

- 웹에 공개된 연구 결과에 따르면, 29% 가량의 웹 페이지 콘텐츠는 중복이다. 따라서 같은 콘텐츠를 여러 번 저장하게 될 수 있다.
- 이 문제를 해결하기 위한 자료 구조를 도입하여 데이터 중복을 줄이고 데이터 처리에 소요되는 시간을 줄인다.
- 두 HTML 문서를 비교하는 가장 간단한 방법은 그 두 문서를 문자열로 보고 비교하는 것이겠지만, 비교 대상 문서의 수가 10억에 달하는 경우에는 느리고 비효율적일 것이다.
- 효과적인 방법은 웹 페이지의 해시 값을 비교하는 것이다.

콘텐츠 저장소

- 콘텐츠 저장소는 HTML 문서를 보관하는 시스템이다. 저장소를 구현하는데 쓰일 기술을 고를 때는 저장할 데이터의 유형, 크기, 저장소 접근 빈도, 데이터의 유효 기간 등을 종합적으로 고려해야 한다.
    - 데이터 양이 너무 많으므로 대부분의 콘텐츠는 디스크에 저장한다.
    - 인기 있는 콘텐츠는 메모리에 두어 접근 지연시간을 줄일 것이다.

URL 추출기

- URL 추출기는 HTML 페이지를 파싱하여 링크들을 골라내는 역할을 한다.
- 상대 경로(relative path)는 전부 절대 경로(absolute Path)로 변환한다.

URL 필터

- URL 필터는 특정한 콘텐츠 타입이나 파일 확장자를 갖는 URL, 접속 시 오류가 발생하는 URL, 접근 제외 목록(deny list)에 포함된 URL 등을 크롤링 대상에서 배제하는 역할을 한다.

이미 방문한 URL?

- 이미 방문한 URL이나 미수집 URL 저장소에 보관된 URL을 추적할 수 있도록 하는 자료 구조를 사용할 것이다.
- 이미 방문한 적이 있는 URL인지 추적하면 같은 URL을 여러번 처리하는 일을 방지할 수 있다 (서버 부하를 줄일고 시스템이 무한 루프에 빠지는 일을 방지한다)
- 해당 구조는 블룸 필터(bloom filter)나 해시 테이블이 널리 쓰인다.

웹 크롤러 작업 흐름

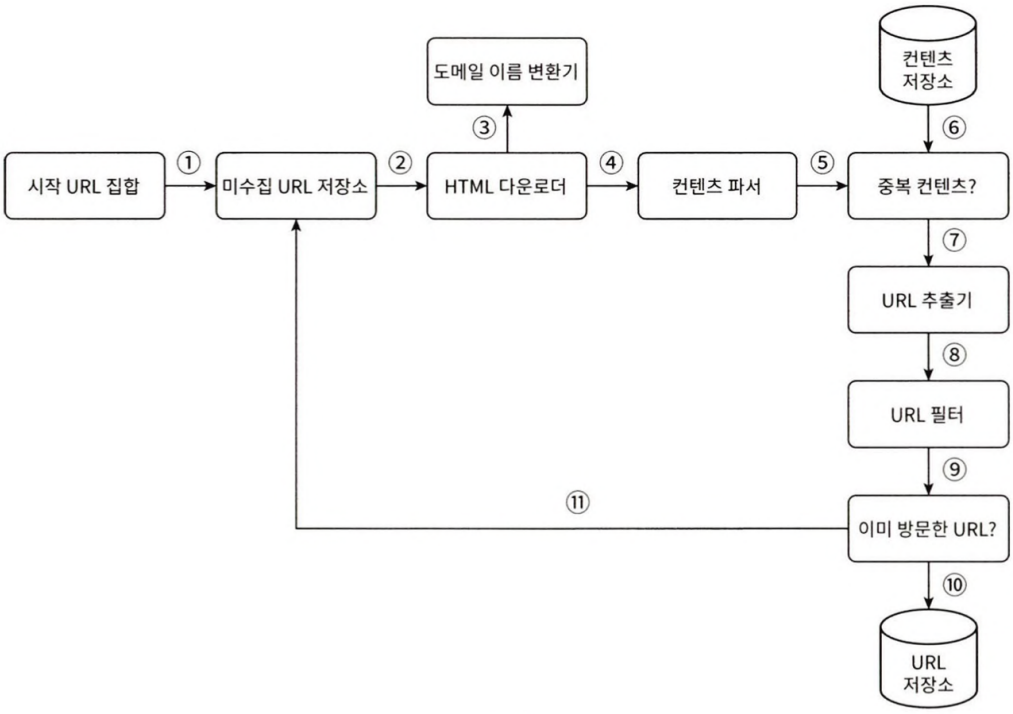


그림 9-4

1. 시작 URL들을 미수집 URL 저장소에 저장한다.
2. HTML 다운로더는 미수집 URL 저장소에서 URL 목록을 가져온다.

3. HTML 다운로더는 도메인 이름 변환기를 사용하여 URL의 IP 주소를 알아내고, 해당 IP 주소로 접속하여 웹 페이지를 다운받는다.
4. 콘텐츠 파서는 다운된 HTML 페이지를 파싱하여 올바른 형식을 갖춘 페이지인지 검증한다.
5. 콘텐츠 파싱과 검증이 끝나면 중복 컨텐츠인지 확인하는 절차를 개시한다.
6. 중복 콘텐츠인지 확인하기 위해서, 해당 페이지가 이미 저장소에 있는지 본다.
    - 이미 저장소에 있는 콘텐츠인 경우에는 처리하지 않고 버린다.
    - 저장소에 없는 콘텐츠인 경우에는 저장소에 저장한 뒤 URL 추출기로 전달한다.
7. URL 추출기는 해당 HTML 페이지에서 링크를 골라낸다.
8. 골라낸 링크를 URL 필터로 전달한다.
9. 필터링이 끝나고 남은 URL만 중복 URL 판별 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 뿐 아니라 미수집 URL 저장소에도 전달한다.

상세 설계

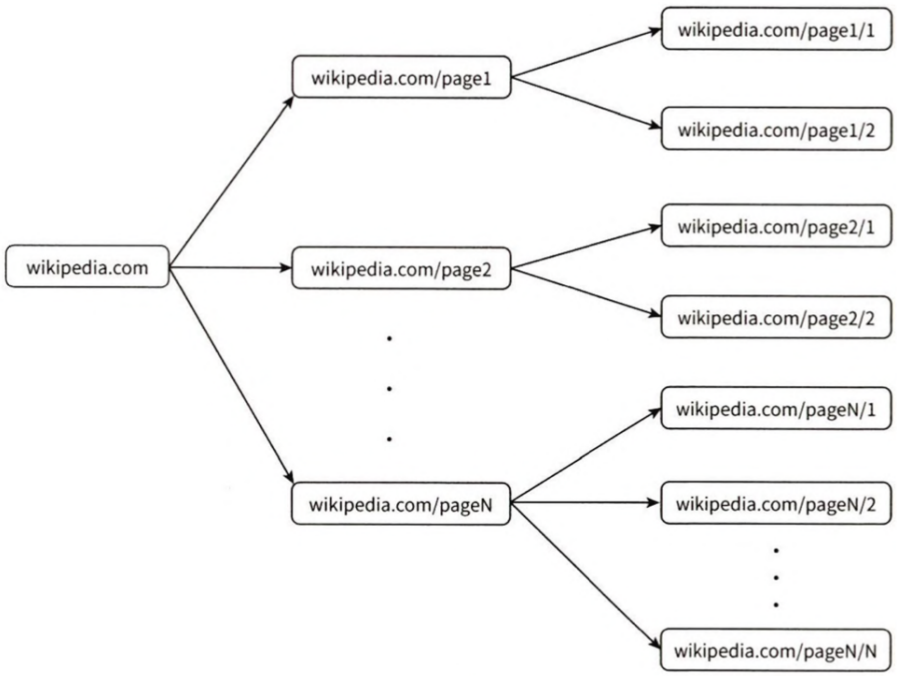
- 중요 컴포넌트 소개
    1. DFS(Depth-First Search) vs BFS(Breath-First Search)
    2. 미수집 URL 저장소
    3. HTML 다운로더
    4. 안정성 확보 전략
    5. 확장성 확보 전략
    6. 문제 있는 컨텐츠 감지 및 회피 전략

DFS를 쓸 것인가, BFS를 쓸 것인가

- 웹은 유향 그래프(Directed Graph)나 같다. 페이지는 노드이고, 하이퍼링크(URL)는 에지(edge)라고 본다.
- 크롤링 프로세스는 이 유향 그래프를 에지를 따라 탐색하는 과정이다.
- DFS, BFS는 바로 이 그래프 탐색에 널리 사용되는두 가지 알고리즘이다.
    - DFS, 즉 깊이 우선 탐색법(depth-first search)은 좋은 선택이 아닐 가능성이 높다.
    - 웹 크롤러는 보통 BFS, 즉 너비 우선 탐색법(breath-first search)을 사용한다.
    - BFS는 FIFO(FIrst-In-First-Out) 큐를 사용하는 알고리즘이다.

BFS 문제점 - 1

- 한 페이지에서 나오는 링크의 상당수는 같은 서버로 되돌아간다.
- 크롤러는 같은 호스에 속한 많은 링크를 다운로드 받느라 바빠지게 되는데, 이때 이링크들을 병렬로 처리한다면 수많은 요청에 의해 크롤링 하는 서버가 과부하를 받게된다.



- 보통 이런 크롤러는 ‘예의 없는(inpolite)’ 크롤러로 간주된다.

BFS 문제점 -2

- 표준 BFS 알고리즘은 URL 간에 우선순위를 두지 않는다.

→ 처리 순서에 있어 공평함

- 모든 웹 페이지가 같은 수준의 품질, 같은 수준의 중요성을 갖지 않는다.
- 페이지 순위(page rank), 사용자 트래픽의 양, 업데이트 빈도 등 여러 가지 척도에 비추어 처리 우선순위를 구별하는 것이 온당하다.

미수집 URL 저장소

- 미수집 URL 저장소를 활용하면 이런 문제를 좀 쉽게 해결이 가능하다.

  → URL 저장소는 다운로드할 URL을 보관하는 장소다.

- 이 저장소를 잘 구현하면 '예의(Politeness)'를 갖춘 크롤러, URL 사이의 우선순위와 신선도(freshness)를 구별하는 크롤러를 구현할 수 있다.

예의

- 웹 크롤러는 수집 대상 서버로 짧은 시간 안에 너무 많은 요청을 보내는 것을 삼가해야 한다.
- 너무 많은 요청을 보내는 것은 '무례한(impolite)' 일이며, 때로는 Dos(Denial-of-Service) 공격으로 간주되기도 한다.
- 예의 바른 크롤러를 만드는 데 있어서 지켜야할 한 가지 원칙은, 동일 웹 사이트에 대해서는 한번에 한 페이지만 요청을 한다는 것이다.
- 웹 사이트의 호스트명(hostname)과 다운로드를 수행 작업 스레드(worker thread) 사이의 관계를 유지하면 된다.
  - 각 다운로드 스레드는 별도 FIFO 큐를 가지고 있어서, 해당 큐에서 꺼낸 URL만 다운로드를 한다.

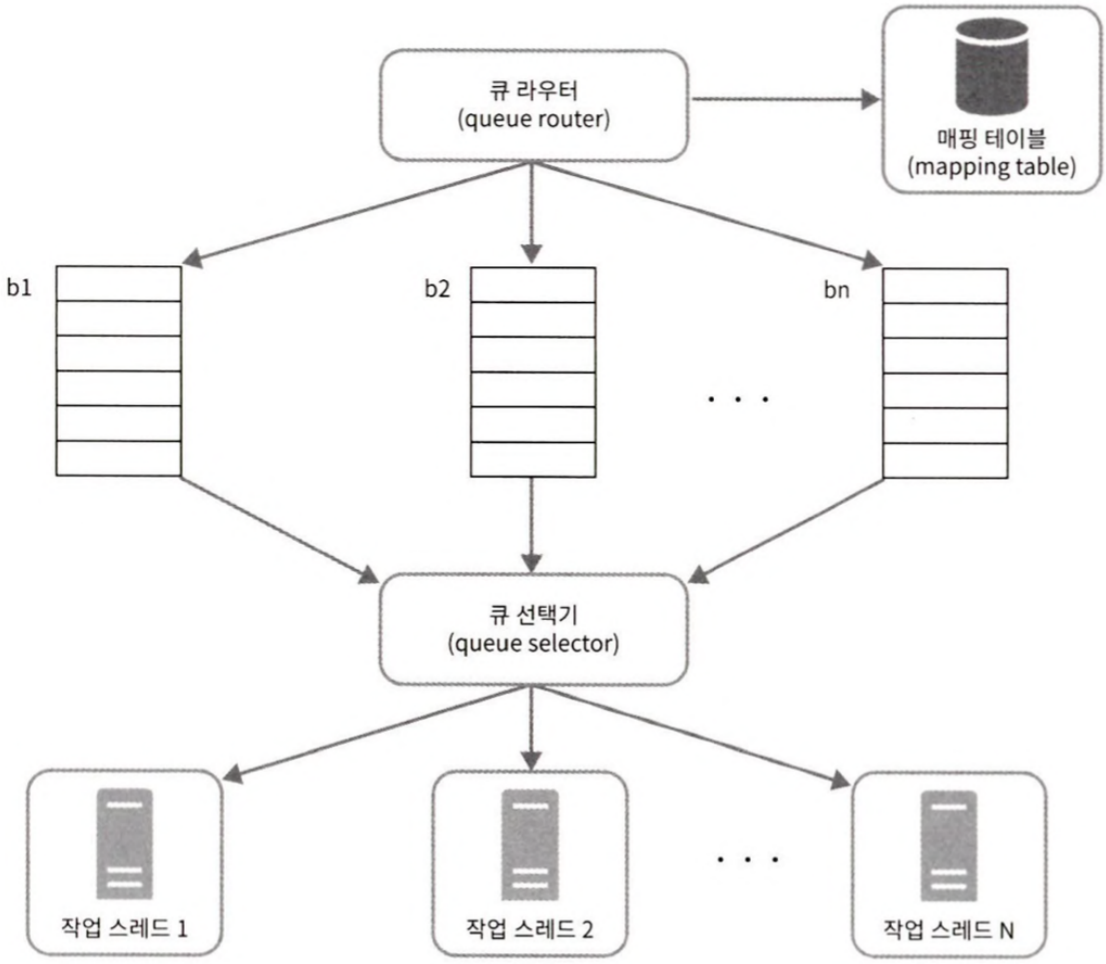


그림 9-6

큐 라우터(queue router)

- 같은 호스트에 속한 URL은 언제나 같은 큐(b1, b2, …, bn)로 가도록 역할을 한다.

매핑 테이블(mapping table)

- 호스트 이름과 큐 사이의 관계를 보관하는 테이블.

FIFO 큐(b1부터 bn까지)

- 같은 호스트에 속한 URL은 언제나 같은 큐에 보관한다.

큐 선택기(queue selector)

- 큐 선택기는 규들을 순회하면서 큐에서 URL을 꺼내서 해당 큐에서 나온 URL을 다운로드하도록 지정된 작업 스레드에 전달하는 역할을 한다.

작업 스레드(worker thread)

- 작업 스레드는 전달된 URL을 다운로드하는 작업을 수행한다.
- 전달된 URL은 순차적으로 처리될 것이며, 작업들 사이에는 일정한 지연시간(delay)을 둘 수 있다.

우선순위

- 애플(apple) 제품에 대한 사용자 의견이 올라오는 포럼의 한 페이지가 애플 홈페이지와 같은 중요도를 갖는다고는 보기는 어려울 것이다.
- 유용성에 따라 URL의 우선운쉬를 나눌 때는 페이지랭크(PageRank), 트래픽 양, 갱신 빈도(update frequency) 등 다양한 척도를 사용할 수 있을 것이다.
- 순위결정장치(prioritizer)는 URL 우선순위를 정하는 컴포넌트다.

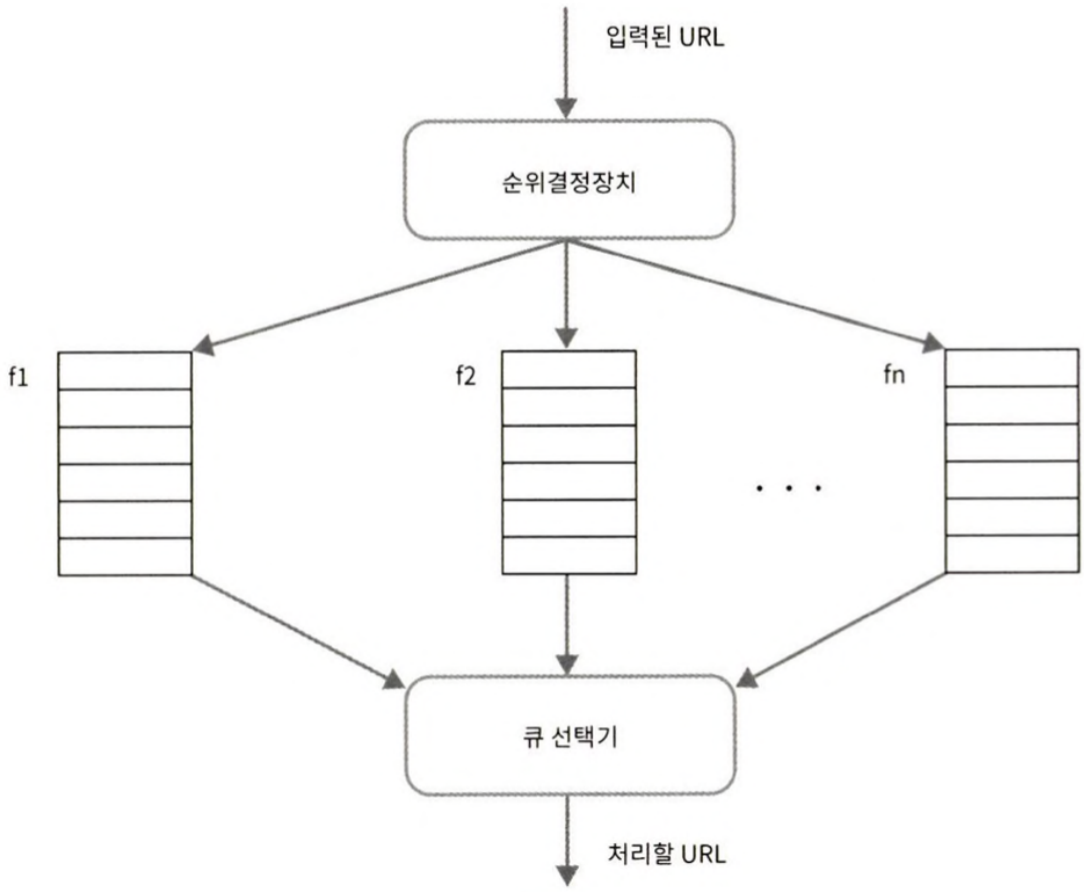


그림 9-7

순위 결정 장치(prioritizer)

- URL을 입력으로 받아 우선순위를 계산한다.

큐(f1, … fn)

- 우선순위별로 큐가 하나씩 할당된다. 순위가 높으면 선택될 확률도 높아진다.

큐 선택기

- 임의 큐에서 처리할 URL을 꺼내는 역할을 담당한다. 순위가 높은 큐에서 더 자주 꺼내도록 프로그램되어 있다.

전면 큐 (front queue)

- 우선순위 결정 과정을 처리한다.

후면 큐 (back queue)

- 크롤러가 예의 바르게 동작하도록 보증한다.

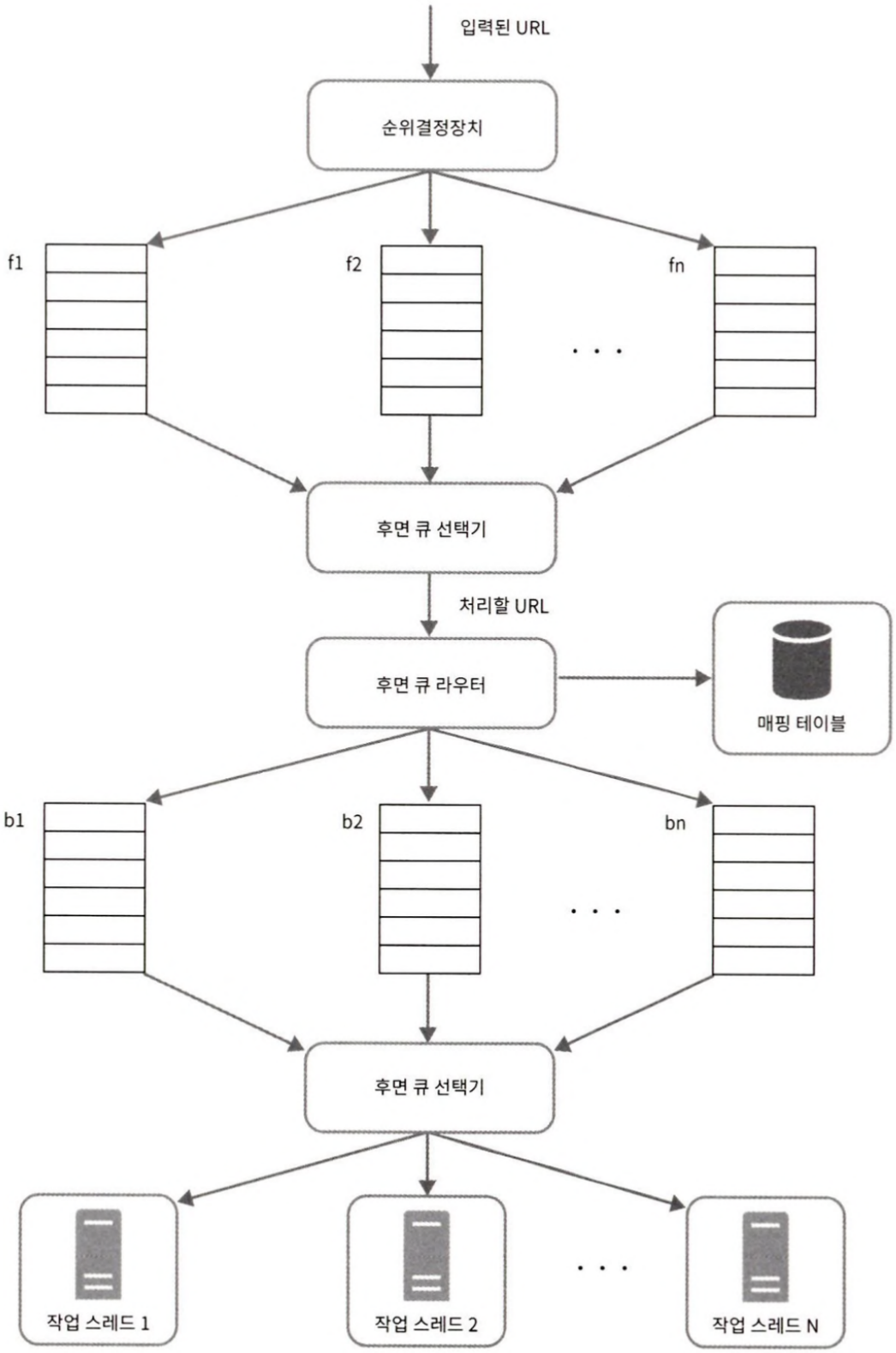

입력된 URL
순위결정장치
f1
f2
fn
후면 큐 선택기
처리할 URL
후면 큐 라우터
매핑 테이블
b1
b2
bn
후면 큐 선택기
작업 스레드 1
작업 스레드 2
작업 스레드 N

그림 9-8

신선도

- 웹 페이지는 수시로 추가되고, 삭제되고, 변경된다.
- 데이터의 신선함(freshness)을 유지하기 위해서는 이미 다운로드한 페이지라도 주기적으로 재수집(recrawl)할 필요가 있다.
    - 웹 페이지 변경 이력(update history) 활용
    - 우선순위를 활용하여, 중요한 페이지는 좀 더 자주 재수집

미수집 URL 저장소를 위한 지속성 저장장치

- 검색 엔진을 위한 크롤러의 경우, 처리해야 하는 URL의 수는 수억 개에 달한다.
- 모두를 메모리에 보관하는 것은 안정성이나 규모 확장성 측면에서 바람직하지 않다.
- 전부 디스크에 저장하는 것도 좋은 방법은 아닌데, 느려서 쉽게 성능 병목지점이 되기 때문이다.
- 설계안에 따라 절충안(hybrid approach)을 택한다
    - URL은 디스크에 두지만 IO 비용을 줄이기 위해 메모리 버퍼에 큐를 두는 것이다.
    - 버퍼에 있는 데이터는 주기적으로 디스크에 기록할 것이다.

HTML 다운로더

- HTTP 프로토콜을 통해 웹 페이지를 내려받는다.
- 로봇 제외 프로토콜(Robot Exclusion Protocol)부터 살펴보자

Robots.txt

- 로봇 제외 프로토콜이라고 부르기도 하는 Robots.txt는 웹사이트가 크롤러와 소통하는 표준적 방법이다.
- 크롤러가 수집해도 되는 페이지 목록이 들어있다.
- 웹 사이트를 긁어 가기 전에 크롤러는 해당 파일에 나열된 규칙을 확인해야 한다.

```
User-agent: Googlebot
Disallow: /creatorhub/*
```

```
Disallow: /rss/people/*/reviews
Disallow: /gp/pdp/rss/*/reviews
Disallow: /gp/cdp/member-reviews/
Disallow: /gp/aw/c r/
```

- creatorhub 같은 디렉터리의 내용은 다운받을 수 없다는 것을 알 수 있다.

성능 최적화

1. 분산 크롤링

- 성능을 높이기 위해 크롤링 작업을 여러 서버에 분산하는 방법이다.
- 각 서버는 여러 스레드를 돌려 다운로드 작업을 처리한다.
- 이 구성을 위해 URL 공간은 작은 단위로 분할하여, 각 서버는 그중 일부의 다운로드를 담당하도록 한다.

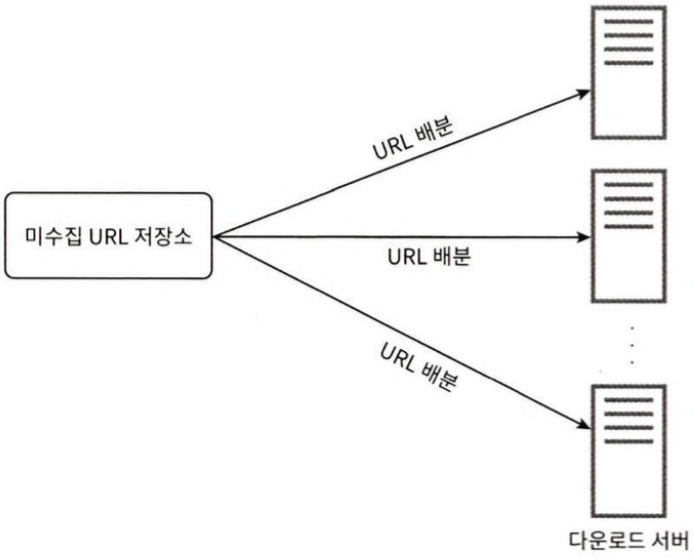


그림 9-9

2. 도메인 이름 변환 결과 캐시

- 도메인 이름 변환기(DNS Resolver)는 크롤러 성능의 병목 중 하나인데, 이는 DNS 요청을 보내고 결과를 받는 작업의 동기적 특성 때문이다.
- DNS 요청이 처리되는 데는 보통 10ms에서 200ms가 소요된다.

- 크롤러 스레드 가운데 어느 하나라도 이 작업을 하고 있으면 다른 스레드의 DNS 요청은 전부 블록(block)된다.
- DNS 조회 결과로 얻어진 도메인 이름과 IP 주소 사이의 관계를 캐시에 보관해 놓고 크론 잡(cron Job) 등을 돌려 주기적으로 갱신하도록 해 놓으면 성능을 효과적으로 높일 수 있다.

3. 지역성

- 크롤링 작업을 수행하는 서버를 지역별로 분산하는 방법이다.
- 크롤링 서버가 크롤링 대상 서버와 지역적으로 가까우면 페이지 다운로드 시간은 줄어들 것이다.
- 지역성(locality)을 활용하는 전략은 크롤 서버, 캐시, 큐, 저장소 등 대부분의 컴포넌트에 적용 가능하다.

4. 짧은 타임아웃

- 어떤 웹 서버는 응답이 느리거나 아예 응답하지 않는다. 이런 경우에 대기 시간(wait time)이 길어지면 좋지 않으므로, 최대 얼마나 기다릴지를 미리 정해두는 것이다.
- 이 시간 동안 서버가 응답하지 않으면 크롤러는 해당 페이지 다운로드를 중단하고 다음 페이지로 넘어간다.

안정성

- 최적화된 서능뿐 아니라 안정성도 다운로더 설계 시 중요하게 고려해야 할 부분이다.
- 시스템 안정성을 향상시키기 위한 접근법 가운데 중요한 몇 가지는 아래와 같다.
  - 안정 해시(consistent hashing)
    - 다운로더 서버들에 부하를 분산할 때 적용이 가능한 기술
  - 크롤링 상태 및 수집 데이터 저장
    - 장애가 발생한 경우에도 쉽게 복구할 수 있도록 크롤링 상태와 수집된 데이터를 지속적 저장장치에 기록해 두는것
  - 예외 처리(exception handling)
    - 대규모 시스템에서 에러(error)는 흔하게 발어진다.

        - 예외가 발생해도 전체 시스템이 중단되는 일은 없고 우아하게 이어나가야 한다.
    - 데이터 검증(data validation)
        - 시스템 오류를 방지하기 위한 중요 수단 가운데 하나이다.

확장성

- 진화하지 않는 시스템은 없는 법이다.
- 새로운 형태의 콘텐츠를 쉽게 지원할 수 있도록 신경써야 한다.
- PNG 다운로더는 PNG 파일을 다운로드하는 플러그인이다.
- 웹 모니터(web monitor)는 웹을 모니터링하여 저작권이나 상표권이 침해되는 일을 막는 모듈이다.

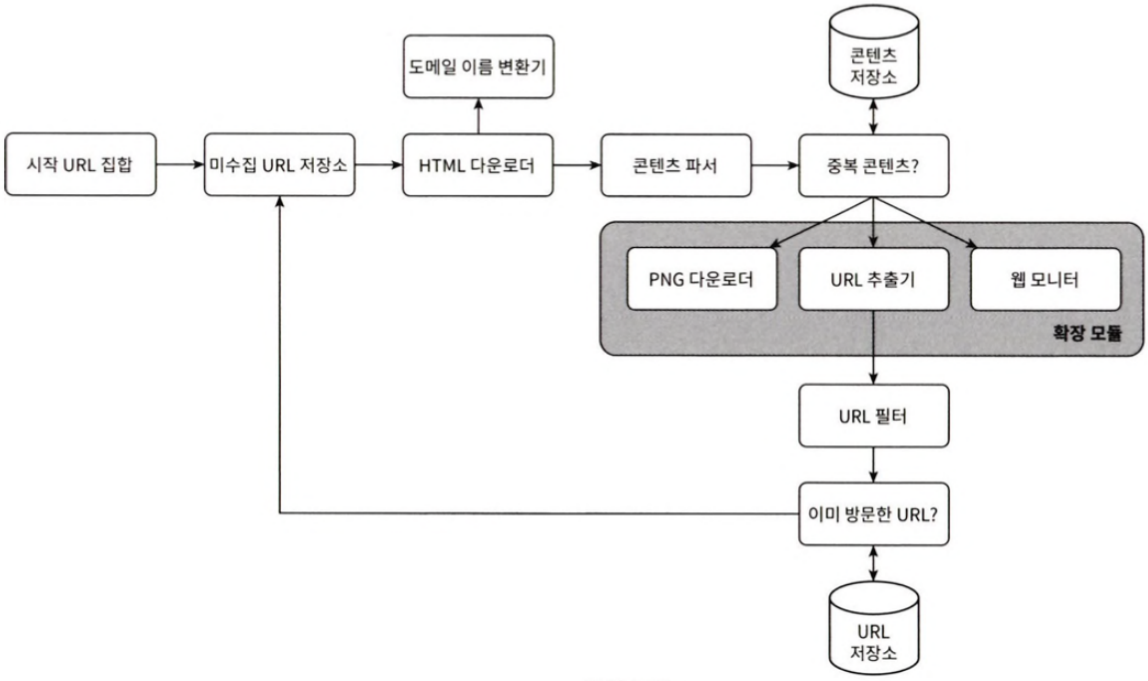


그림 9-10

문제 있는 콘텐츠 감지 및 회피

- 중복이거나 의미 없는, 또는 유해한 콘텐츠를 감지하는 시스템

1. 중복 컨텐츠

- 해시나 체크섬(check-sum)을 사용하면 중복 콘텐츠를 보다 쉽게 탐지할 수 있다.

2. 거미 덫

- 거미 덫(spider trap)은 크롤러를 무한 루프에 빠드리도록 설계한 웹 페이지다.

  `spidertrapexample.com/foo/bar/foo/bar/foo/bar/...`

- 이런 덫은 URL의 최대 길이를 제한하면 피할 수 있다.
- 덫을 자동으로 피해가는 알고리즘을 만들어내는 것은 까다롭다.
- 한 가지 방법은 사람이 수작업으로 덫을 확인하고 찾아낸 후에 덫이 있는 사이트를 크롤러 탐색 대상에서 제외하거나 URL 필터 목록에 걸어두는 것이다.

3. 데이터 노이즈

- 어떤 콘텐츠는 거의 가치가 없다. 광고나 스크립트 코드, 스팸 URL 같은 것이 그렇다.
- 크롤러에게 도움이 될 것이 없으므로 가능하다면 제외해야 한다.

마무리

- 좋은 크롤러가 갖추어야하는 특성을 살펴보았다.
  - 규모 확장성(scalability)
  - 예의
  - 확장성(extensibility)
  - 안정성
- 규모 확정성이 뛰어난 웹 크롤러 설계 작업은 단순하지 않다.
- 다음 주제를 추가로 논의해보면 좋을 것이다.
  - 서버 측 렌더링(server-side rendering) : 많은 사이트가 자바스크립트 기술을 사용해 링크를 적석에서 만들어 낸다. 웹 페이지를 그냥 있는 그대로 다운받아 파싱해보면 그렇게 동적으로 생성되는 링크는 발견할 수 없을 것이다. 페이지 파싱전 서버 측 렌더링(동적 렌더링dynamic rendering)을 적용하면 해결할 수 있다.

- 원치 않은 페이지 필터링: 저장 공간 등 크롤링에 소요되는 자원은 유한하기 때문에 스팸 방지(anti-spam) 컴포넌트를 두어 품질이 조악하거나 스팸 페이지를 걸러내도록 해 두면 좋다.
- 데이터베이스 다중화 및 샤딩: 다중화(replication)나 샤딩(sharding) 같은 기법을 적용하면 데이터 계층(data layer)의 가용성, 규모 확장성, 안정성이 향상된다.
- 수평적 규모 확장성(horizontal scalability): 대규모의 크롤링을 위해서는 다운로드를 실행할 서버가 수백 혹은수천 대 필요하게 될 수도 잇다. 수평적 규모 확장성을 달성하는 데 중요한 것은 서버가 상태정보를 유지하지 않도록 하는 것, 즉 무상태(stateless) 서버로 만드는 것이다.
- 가용성, 일관성, 안정성
- 데이터 분석 솔루션(analytics): 데이터를 수집하고 분석하는 것은 어느 시스템이나 중요하다. 시스템을 세밀히 조정하기 위해서는 이런 데이터와 그 분석 결과가 필수적이다.