# 대용량 시스템 설계 기초

## 9.웹크롤러 설계

@OolongTea620
김혜린

2024.04.07

## 들어가기 전에

**웹 크롤러란?**

**웹페이지, 이미지, 비디오... 파일 등 웹 리소스를 모으는 시스템**

## 들어가기 전에

### 어디에 이용되는가?

검색 엔진 인덱싱(search engine indexing),
웹 아카이빙(Web Archiving),
웹 마이닝(Web Mining),
웹 모니터링(web Monitoring)

## 들어가기 전에

웹 크롤러의 복잡도는 처리해야하는 데이터의 규모에 따라 달라진다.

**설계 전, 웹 크롤러가 감당해야할 데이터의 규모와 기능을 고려해야한다.**

## 1단계 - 문제 이해 및 설계 범위 확정

### 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 URL목록에 추가하고, 위의 과정을 처음부터 반복한다.

## 1단계 - 문제 이해 및 설계 범위 확정

## 웹 크롤러가  만족해야할 속성

**규모 확장성**
웹은 거대하고, 수십억 개의 페이지가 존재.
따라서 병행성(parallelism)을 활용시, 보다 효과적으로 웹 크롤링 가능.

**안정성(robustness)**
크롤러는 이런 비정상적 입력이나 환경에 잘 대응할 수 있어야
(잘못 작성된 HTML, 아무 반응이 없는 서버, 장애, 악성 코드 링크...)

## 1단계 - 문제 이해 및 설계 범위 확정

## 웹 크롤러가  만족해야할 속성

**예절(Politeness)**
**크**롤러는 수집 대상 웹 사이트에 짧은 시간 동안 너무 많은 요청을 보내서는 안된다

**확장성**
새로운 형태의 콘텐츠를 지원하기가 쉬워야
(이미지 파일도 크롤링 하려할 때, 전체 시스템 설계에 영향이 없도록 해야)

## 1단계 - 문제 이해 및 설계 범위 확정

### 개략적 규모 추정

매달 10억 개의 웹 페이지를 다운로드 한다.
QPS = 10억 / 30일 / 24시간 / 3600초 = 대략 400 page / sec
최대 (Peak) QPS = 2 X QPS = 800 page/sec

웹 페이지의 평균 크기는 500k라고 가정
10억 페이지 x 500k = 500 TB /month.

500TB X 12개월 X 5년 = 30PB의 저장용량이 필요하다
(1개월치 데이터를 보관하는 데는 500 TB, 5년간 보관한다고 가정 시)

## 2단계 개략적 설계안 제시 및 동의 구하기

### 시작 URL 집합

**크롤러가 가능한 많은 링크를 탐색할 수 있도록 하는 URL**을 고르는 것이 바람직
전체 URL 공간을 작은 부분 집합으로 나누는 전략 사용

**주제별로 다른 시작 URL을 사용**
URL 공간을 쇼핑, 스포츠, 건강 등등의 주제별로 세분화하고 그 각각에 다른 시작 URL을 쓰는 것

## 2단계 개략적 설계안 제시 및 동의 구하기

### 미수집 URL 저장소

다운로드 할 URL을 저장 관리하는 컴포넌트
FIFO 큐

### HTML 다운로더

인터넷에서 웹 페이지를 다운로드하는 컴포넌트
다운로드할 페이지의 URL은 미수집 URL 저장소가 제공한다.

## 2단계 개략적 설계안 제시 및 동의 구하기

### 도메인 이름 변환기

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

### 콘텐츠 파서

웹 페이지를 **다운로드 전 파싱과 검증 절차를 필요**
크롤링 서버 안에 콘텐츠 파서를 구현하면 크롤링 과정이 느려지게 될 수 있으므로 **독립된 컴포넌트로 구성**

## 2단계 개략적 설계안 제시 및 동의 구하기

### 콘텐츠 저장소

HTML 문서를 보관하는 시스템
저장소를 구현하는 데 쓰일 기술을 고를 때는
저장할 데이터의 유형, 크기, 저장소 접근 빈도, 데이터의 유효 기간 등을 종합적으로 고려해야

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

### URL 추출기

HTML 페이지를 파싱하여 링크들을 골라내는 역할

## 2단계 개략적 설계안 제시 및 동의 구하기

### URL 필터

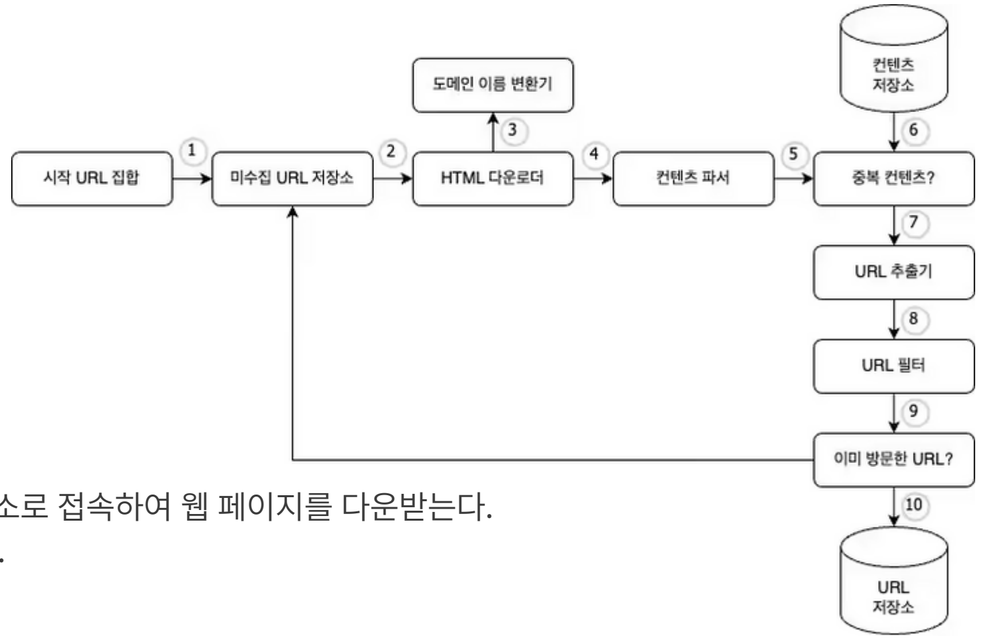
특정한 콘텐츠 타입이나 파일 확장자를 갖는 URL,
접속 시 오류가 발생하는 URL,
접근 제외 목록에 포함된 URL 등을
**크롤링 대상에서 배제하는 역할**

### URL 저장소

이미 방문한 URL을 보관하는 저장소

## 2단계 개략적 설계안 제시 및 동의 구하기

## 웹 크롤러 작업 흐름




1. 시작 URL들을 미수집 URL 저장소에 저장.
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 콘텐츠인지 확인한느 절차를 개시한다.
6. 중복 콘텐츠인지 확인하기 위해서, 해당 페이지가 이미 저장소에 있는지 본다.
    a. 이미 저장소에 있는 콘텐츠인 경우에는 처리하지 않고 버린다.
    b. 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.

## 3단계 상세 설계

## 핵심 컴포넌트 & 구현 기술

- DFS vs BFS
- 미수집 URL 저장소
- HTML 다운로더
- 안정성 확보 전략
- 확장성 확보 전략
- 문제 있는 컨텐츠 감지 및 회피 전략

## 3단계 상세 설계

## DFS 를 쓸 것인가, BFS 를 쓸것인가

웹은 유향 그래프나 같다. 페이지 - 노드, 하이퍼링크 - 엣지.

**크롤링 == 그래프를 탐색하는 과정**

DFS는 좋은 선택이 아닐 가능성이 높다.
> 그래프 크기가 클 경우 어느정도로 **깊숙이 가게 될지 가늠하기 어렵기 때문**.

## 3단계 상세 설계

### 웹 크롤러는 보통 BFS를 사용한다

큐를 사용하는 알고리즘
한쪽으로는 탐색할 URL을 집어넣고, 다른 한 쪽으로는 꺼내기만 한다

### 문제점

**politeness 위배**
한 페이지에서 나오는 링크의 상당수는 같은 서버로 되돌아감

**URL간 우선순위가 없다.**
(처리 순서에 있어 모든 페이지를 공평하게 대우)
페이지 순위, 사용자 트래픽의 양, 업데이트 빈도 등 여러가지 척도에 비추어 구현하려면
표준적 알고리즘만 사용해서는 안됨

---

## 3단계 상세 설계

## 예외

### 크롤러 기본 원칙

동일한 웹사이트에 대해서 **한 번에 한 페이지**

- 같은 페이지를 다운받는 태스크는 시간차를 두고 실행

### 구현 방법

웹사이트의 호스트명과 다운로드를 수행하는 작업 스레드 사이의 관계를 유지한다.

- 각 다운로드 스레드는 별도 FIFO 큐를 가지고 있어, 해당 큐에서 꺼낸 URL만 다운로드 한다

## 3단계 상세 설계

queue router:
같은 호스트에 속한 URL은 언제나 같은 큐로 가도록 보장하는 역할

mapping table:
호스트 이름과 큐 사이의 관계를 보관하는 테이블

FIFO 큐:
같은 호스트에 속한 URL은 언제나 같은 큐에 보관됨

큐 선택기(queue selector):
큐들을 순회하면서 큐에서 URL을 꺼내어
해당 큐에서 나온 URL을 다운로드하도록 지정된 작업스레드에 전달

작업스레드(worker thread):
작업 스레드는 전달된 URL을 다운로드하는 작업을 수행한다.
전달된 URL은 순차적으로 처리될 것이며, 작업들 사이에는 일정한 지연시간을 둘 수 있다

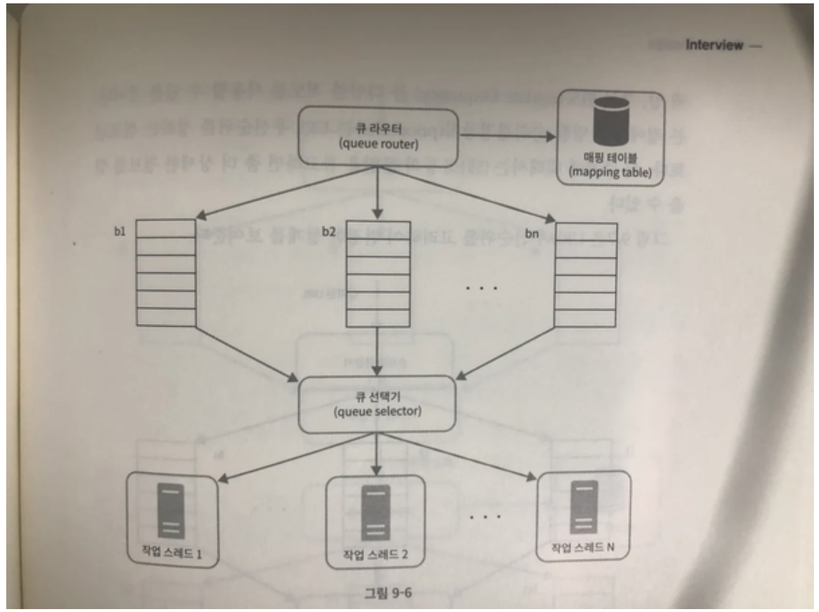


그림 9-6

**호스트별로 들어가는 큐를 지정.**
**같은 큐에서 나온 것은 동시에 접속하지 않는다.**

## 3단계 상세 설계

## 우선 순위

유용성에 따라 URL의 우선순위를 나누는 다양한 컴포넌트
페이지 랭크, 트래픽 양, 갱신 빈도(update frequency) 등 다양한 척도 사용.

### (책에서는) 순위 결정장치(prioritizer) 설명

url 우선순위를 정하는 컴포넌트

1. 순위 결정 장치: URL 을 입력으로 받아 우선순위를 계산.
2. 우선순위 별로 큐가 하나씩 할당됨. 우선순위가 높으면 선택될 확률도 올라감
3. 큐 선택기: 임의 큐에서 처리할 URL을 꺼낸다.
순위가 높은 큐에서 더 자주 꺼내도록 프로그램 되어있음

전면 큐 (front queue): 우선순위 결정과정 처리
후면 큐 (back queue): 크롤러의 politeness 보장.

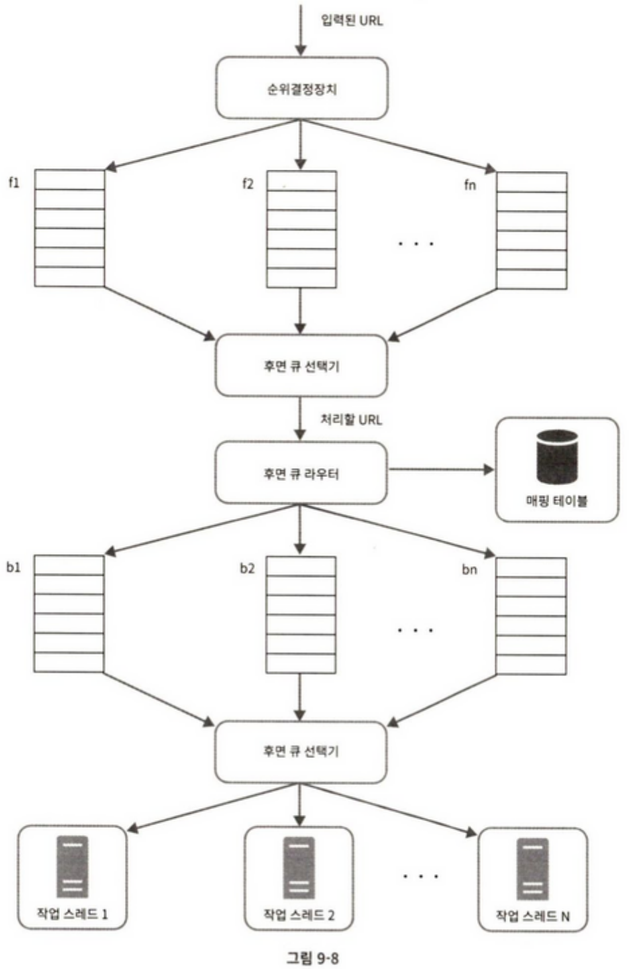


그림 9-8

## 3단계 상세 설계

### 신선도

웹페이지는 수시로 추가/삭제/변경됨
데이터의 신선함(freshness)를 유지하기 위해
이미 다운받은 페이지라 하더라도 재수집 필요 있음,.

### 신선도 작업 최적화 전략

- 웹웹 페이지의 변경 이력 활용
- 우선순위가 높은 페이지는 더 자주 재 수집

## 3단계 상세 설계

### 미수집 URL 저장소를 위한 지속성 저장장치

미수집 URL 저장소를 위한 지속성 저장장치
그 URL들을 어디에 저장해야할까?

**메모리**
❌: 안정성, 규모 확장성이 떨어짐
**디스크**
❌: 느려서 쉽게 성능 병목지점이 됨

**(책에서 설명한)절충안 :**
대부분의 URL은 디스크에 두지만 메모리 버퍼에 큐를 둠.
버퍼에 있는 데이터는 주기적으로 디스크에 기록.

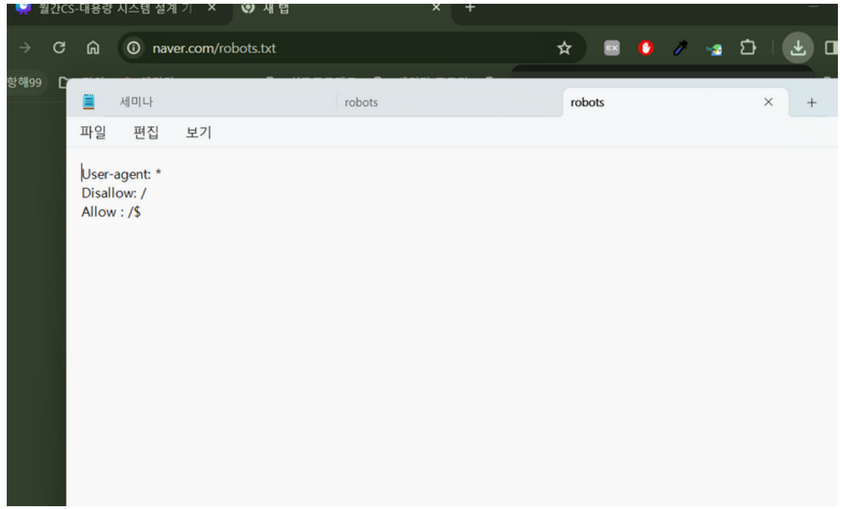
## 3단계 상세 설계

### HTML 다운로더

HTTP 프로토콜을 통해 웹페이지를 내려 받음

### Robots.txt

로봇 제외 프로토콜이라고도 불린다. 웹사이트가 크롤러와 소통하는 표준.




이 파일에는 크롤러가 수집해도 되는 페이지 목록이 들어있다.
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.

## 3단계 상세 설계

## HTML 다운로더의 성능 최적화

**분산 크롤링 :**
성능을 높이기 위해 크롤링 작업을 여러 서버에 분산.

## 3단계 상세 설계

### HTML 다운로더의 성능 최적화

**도메인 이름 변환 결과 캐시 :**

DNS resolver는 크롤러 성능의 병목 중 하나.
이는 DNS 요청을 보내고 결과를 받는 작업의 동기적 특성 때문
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 block됨

DNS 조회 결과로 얻어진 **도메인 이름과 그에 상응하는 IP 주소를 캐시에 보관, 주기적으로 갱신**하도록 구현,
성능을 높일 수 있다.

## 3단계 상세 설계

## HTML 다운로더의 성능 최적화

**지역성:**
크롤링 작업을 수행하는 서버를 지역별로 분산하는 방법
크롤링 서버가 크롤링 대상 서버와 지역적으로 가까우면 페이지 다운로드 시간이 줄어들 것
지역성을 활용하는 전략은 크롤서버, 캐시, 큐, 저장소 등 대부분의 컴포넌트에 적용 가능

**짧은 타임아웃:**
어떤 웹서버가  응답이 느리거나 아예 응답하지 않는 경우를 대비해
크롤러가 최대 얼마나 기다릴지를 미리 정해두는 것.
타임아웃의 경우 해당 페이지 다운로드를 중단한다.

## 3단계 상세 설계

## 안정성

### 안정 해시(Consistent hashing):
다운로더 서버(분산 HTML 다운로더)들에 부하를 고르게 분산하기 위해 적용가능한 기술
- 다운로드 서버를 쉽게 추가하고 삭제 가능

### 크롤링 상태 및 수집 데이터 저장:
장애가 발생한 경우에도 쉽게 복구할 수 있도록 지속적 저장장치에 기록해두는 것
- 크롤링 상태, 수집된 데이터를 지속적으로 저장

## 3단계 상세 설계

## 안정성

**예외처리:**
예외가 발생해도 전체 시스템이 중단되지 않도록 미리 처리

**데이터 검증:**
(비정상정 입력이나 환경을 대비) 시스템 오류를 방지하기 위한 중요 수단.

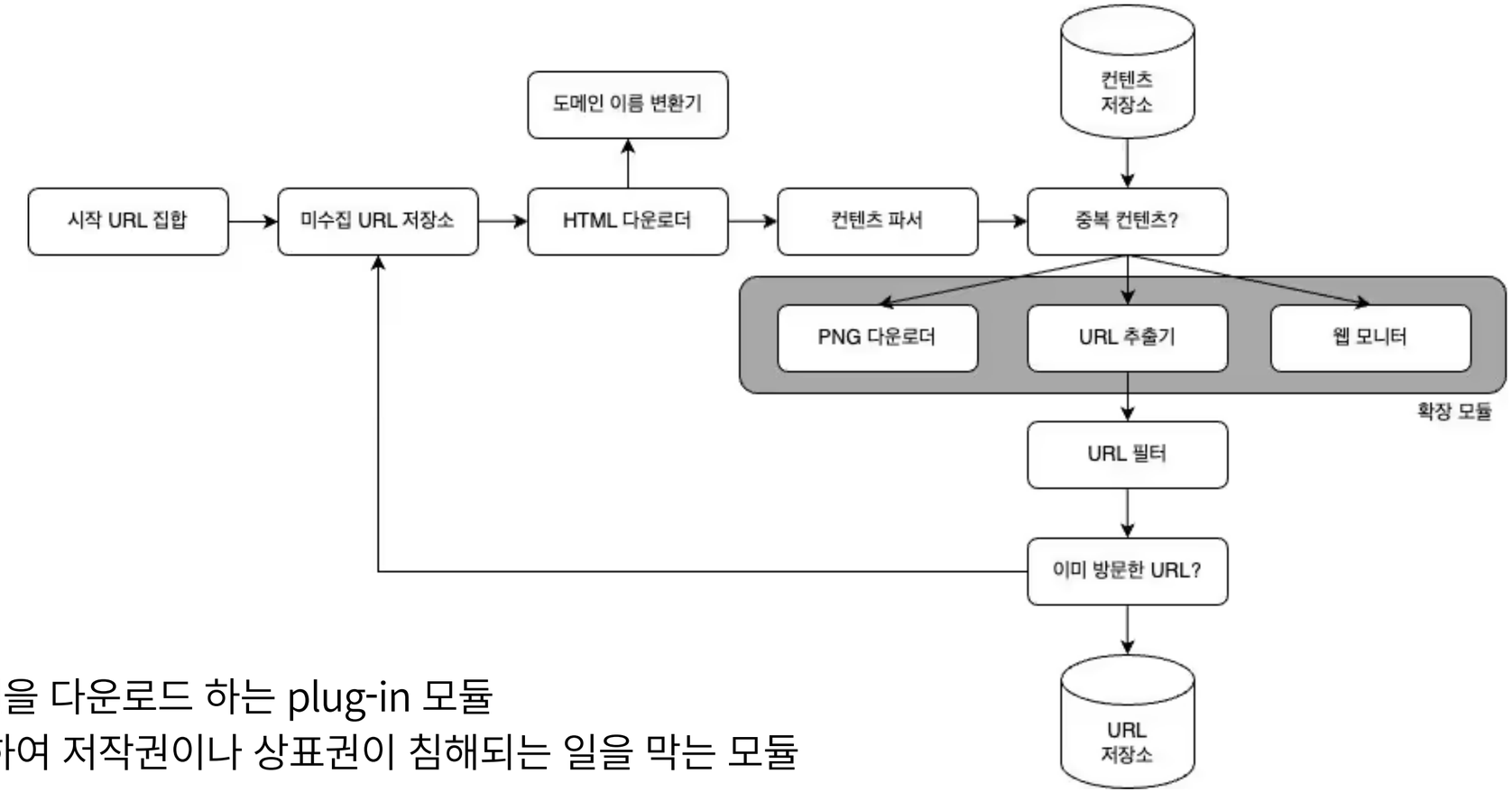
## 3단계 상세 설계

### 확장성

웹 크롤러 시스템의 새로운 형태의 컨텐츠를 쉽게 지원할 수 있도록 신경써야한다.
(본 예제의 경우)새로운 모듈을 끼워 넣음으로써 새로운 형태의 컨텐츠를 지원할 수 있도록 설계

## 3단계 상세 설계

### 확장성




- PNG 다운로더 :  PNG 파일을 다운로드 하는 plug-in 모듈
- 웹 모니터 : 웹을 모니터링하여 저작권이나 상표권이 침해되는 일을 막는 모듈

---

## 3단계 상세 설계

## 문제있는 컨텐츠 감지 및 회피

### 중복 컨텐츠:
웹 컨텐트의 30% 가량은 중복.
해시, 체크섬 등의 방법을 사용해 중복 컨텐츠를 쉽게 탐지 가능
예외가 발생해도 전체 시스템이 중단되지 않도록 미리 처리

### 거미 덫(spider trap):
크롤러를 무한루프에 빠뜨리도록 설계한 웹 페이지

- spidertrapexample.com/foo/bar/foo/bar/foo/bar/...

URL의 최대 길이를 제한(100% 피할 수 없음)
사람이 수작업으로 찾아낸 후 이런 사이트를 크롤러 탐색 대상에서 제외하거나 URL 필터 목록에 걸어둔다.

## 3단계 상세 설계

## 문제있는 컨텐츠 감지 및 회피

**데이터 노이즈:**
어떤 컨텐츠는 거의 가치가 없다. (광고, 스크립트 코드 등) 이런 컨텐츠를 가능한 제외해야

## 4단계 마무리

### 좋은 크롤러가 갖추어야하는 특성

규모 확장성이 뛰어난 웹 크롤러 설계 작업은 단순하지 않다.

- 규모 확장성(scalability)
- 예의(politeness)
- 확장성(extensibility)
- 안정성

## 4단계 마무리

## 추가로 논의해보면 좋은 것

### 서버 측 렌더링 (server-side rendering):

많은 웹사이트가 자바스크립트, ajax 등의 기술을 사용해서 링크를 즉석에서 만들어냄
그러나 웹 페이지를 그냥 있는 그대로 다운받아서 파싱해보면 그렇게 동적으로 생성되는 링크는 발견할 수 없을 것.
이 문제는 페이지를 파싱 전, **동적 렌더링(dynamic rendering)을 적용**하면 해결 가능

### 원치 않는 페이지 필터링:

**스팸 방지 컴포넌트**를 두어 품질이 조약하거나 스팸성인 페이지를 걸러내도록 한다.

## 4단계 마무리

## 추가로 논의해보면 좋은 것

**데이터베이스 다중화 및 샤딩 :**
데이터 계층 가용성, 규모확장성, 안정성

**수평적 규모 확장성:**
대규모 분산 다운로드 서버로 확장하기 위해 무상태 서버로 만들기

**가용성, 일관성, 안정성**
데이터 분석 솔루션(analytics)

# END