Whirlpool

Data Acquisition using N-node Distributed Web Crawler

Rihan Pereira, MSCS

Advisor: Dr. Michael Soltys
Department of Computer Science
MSCS Graduate 2018-2019

November 30, 2019

- Motivation & Contributions

- Motivation & Contributions
- Crawler characteristics & history

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles

} background

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles

} background

- Whirlpool: Event driven architecture

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles

} background

- Whirlpool: Event driven architecture
- Whirlpool: Fetcher

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles

} background

- Whirlpool: Event driven architecture
- Whirlpool: Fetcher
- Whirlpool: Parser

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles

} background

- Whirlpool: Event driven architecture
- Whirlpool: Fetcher
- Whirlpool: Parser
- Whirlpool: Deduplication

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles

} background

- Whirlpool: Event driven architecture
- Whirlpool: Fetcher
- Whirlpool: Parser
- Whirlpool: Deduplication
- Whirlpool: Distributed Crawling

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles

} background

- Whirlpool: Event driven architecture
- Whirlpool: Fetcher
- Whirlpool: Parser
- Whirlpool: Deduplication
- Whirlpool: Distributed Crawling
- Whirlpool: Operations

} implementation & results

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles

} background

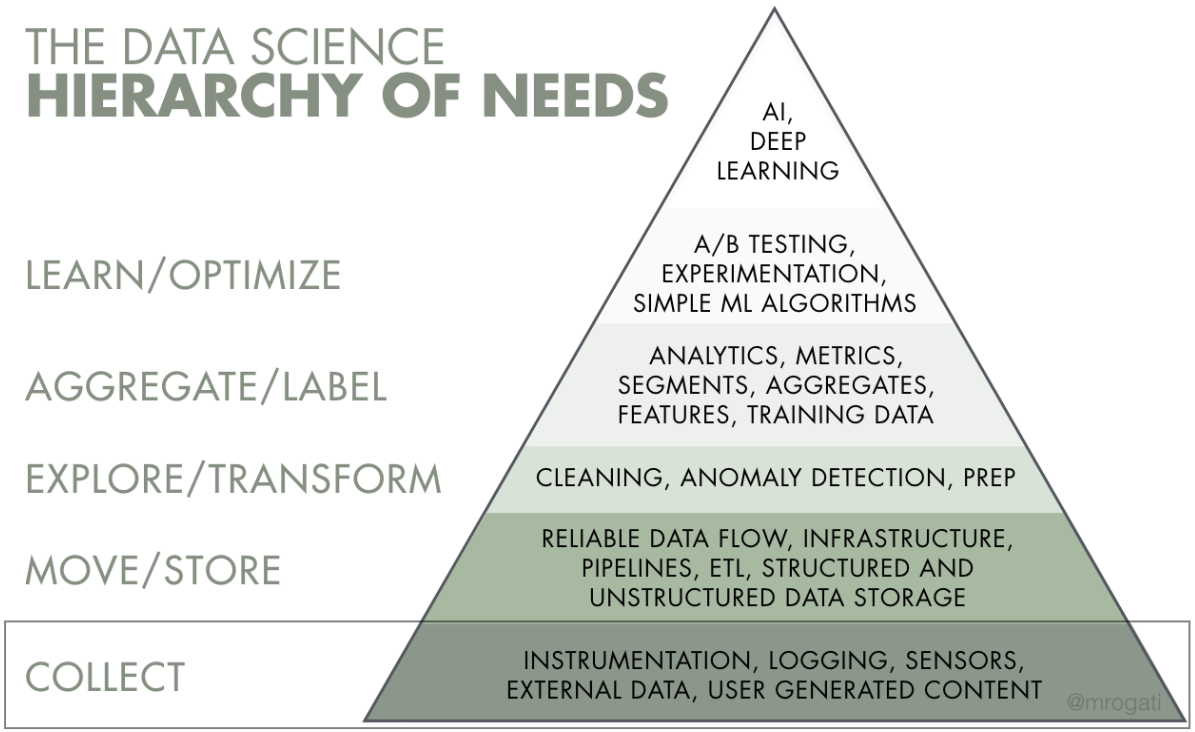
- Whirlpool: Event driven architecture
- Whirlpool: Fetcher
- Whirlpool: Parser
- Whirlpool: Deduplication
- Whirlpool: Distributed Crawling
- Whirlpool: Operations

} implementation & results

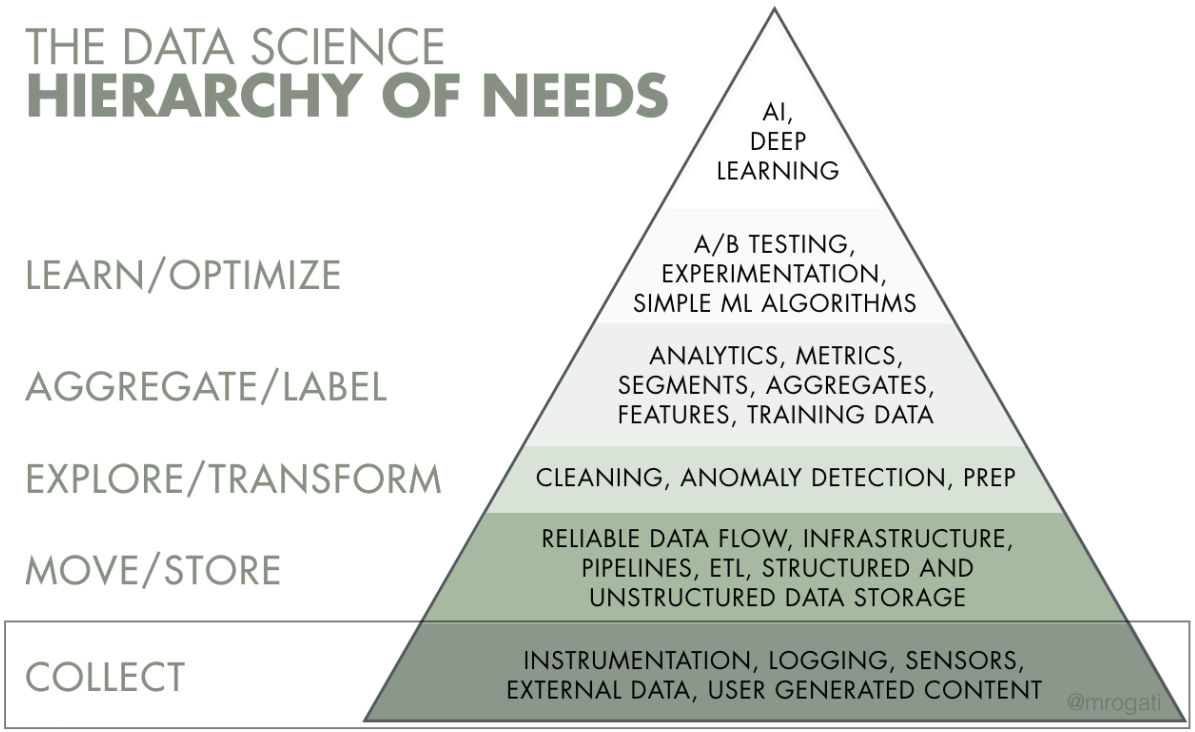
- Future work

Motivation & Contribution

Motivation


THE DATA SCIENCE
HIERARCHY OF NEEDS
AI, DEEP LEARNING
A/B TESTING, EXPERIMENTATION, SIMPLE ML ALGORITHMS
LEARN/OPTIMIZE
ANALYTICS, METRICS, SEGMENTS, AGGREGATES, FEATURES, TRAINING DATA
AGGREGATE/LABEL
CLEANING, ANOMALY DETECTION, PREP
EXPLORE/TRANSFORM
RELIABLE DATA FLOW, INFRASTRUCTURE, PIPELINES, ETL, STRUCTURED AND UNSTRUCTURED DATA STORAGE
MOVE/STORE
INSTRUMENTATION, LOGGING, SENSORS, EXTERNAL DATA, USER GENERATED CONTENT
COLLECT
@mrogati

Motivation



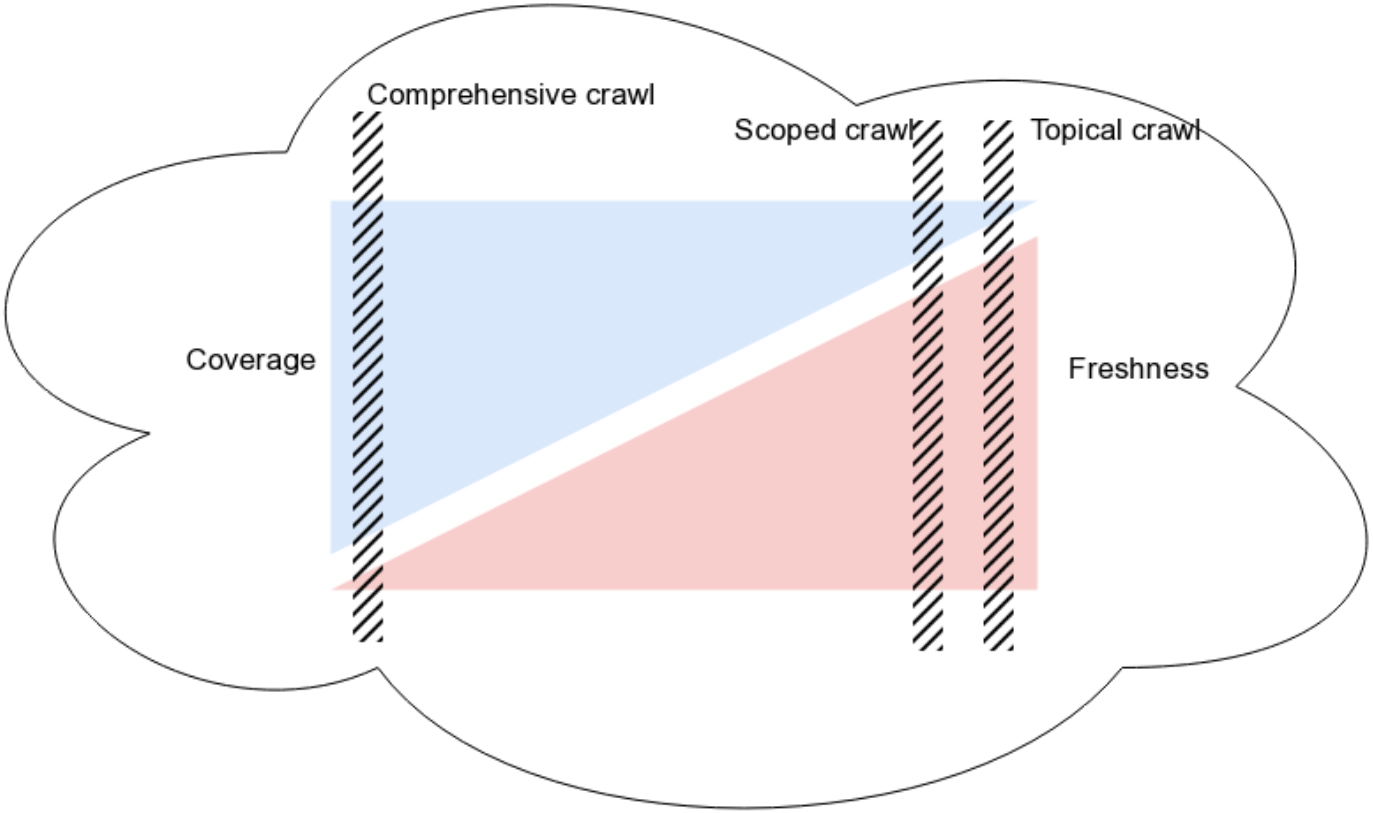
Self-actualization (AI) is great, but you first need food, water, and shelter (data literacy, collection, and infrastructure).”

Contributions

to be completed

Crawler characteristics & history

Coverage & Freshness



Seedlist


MONSTER


glassdoor

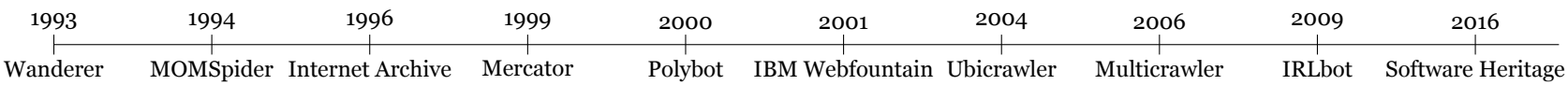

Dice


indeed
one search. all jobs.


ZipRecruiter


SimplyHired

Web crawlers (1990 - 2019)



Mercator 1999 (Heydon & Najork)

basic crawling algorithm

1: Let I $\leftarrow$ {1,2,3,4,5} such that seed set S = $\{U_i |$ i $\in$ I$\}$
2: $U_f \leftarrow$ S; where U_f is a Frontier queue
3: **procedure** Spider(U_f)
4: **while** $U_f \neq \emptyset$ **do**
5: $u \leftarrow \text{Pop}(U_f)$
6: $p \leftarrow \text{Fetch}(u)$
7: $T \leftarrow \exists p$ [{Extract(p, t) | t is a text }]
8: $L \leftarrow \exists p$ [{Extract(p, l) | l is a link }]
9: $U_f \leftarrow U_f \cup L$
10: $\exists u$ [{Delete(U_f, u) | u is a already fetched URL}]
11: **end**

Mercator background

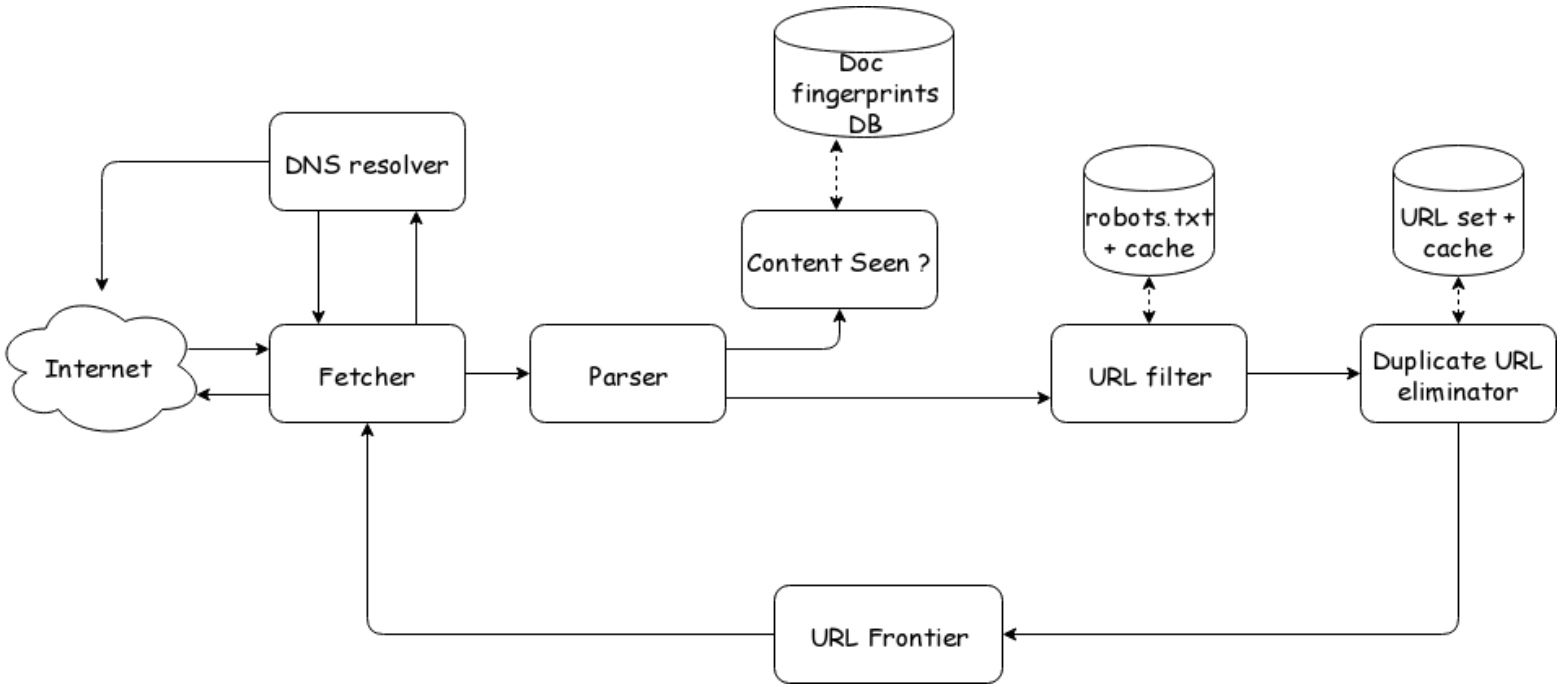
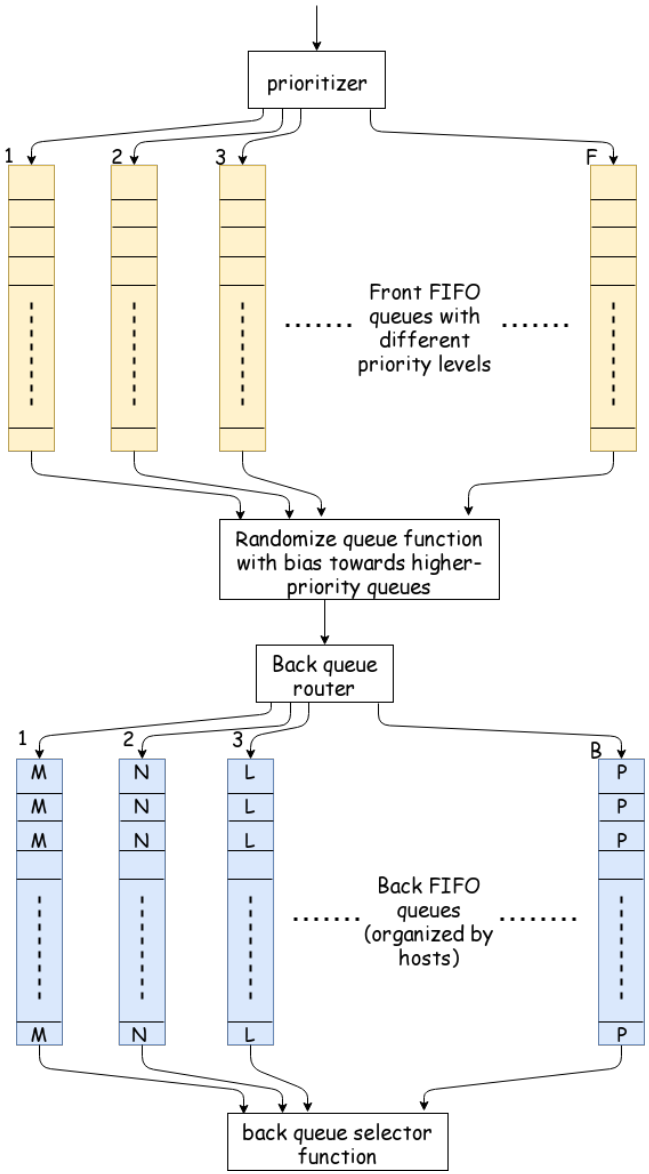
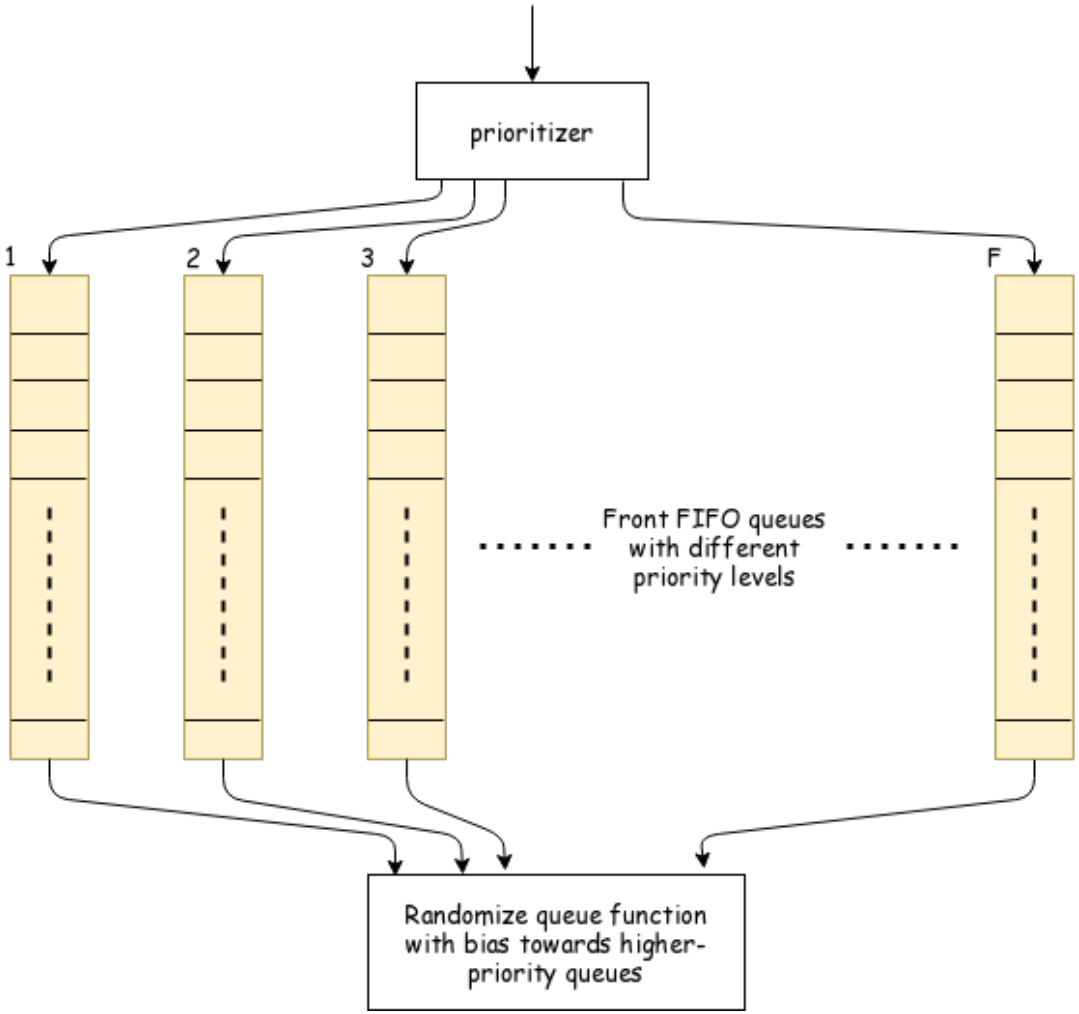


Figure: Mercator building blocks (Heydon & Najork)

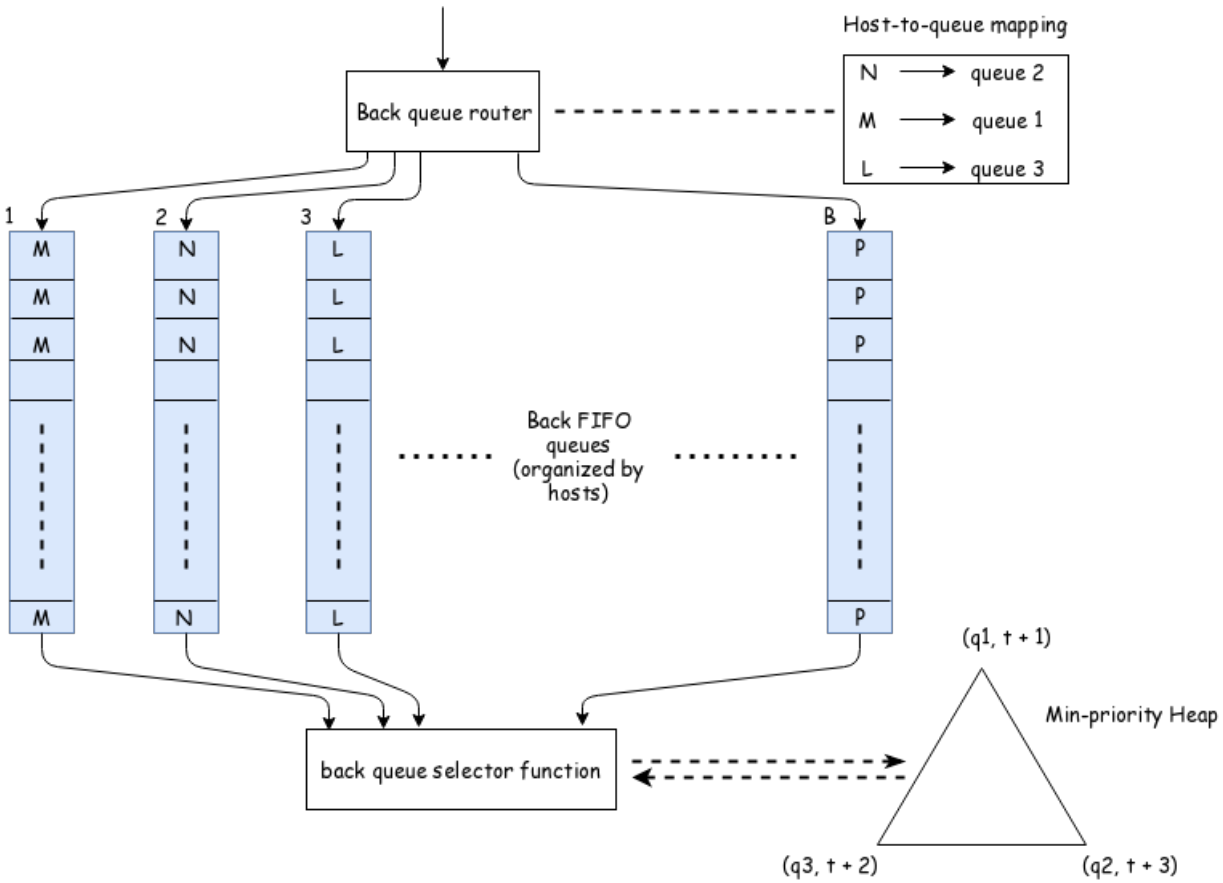
URL Frontier Scheme



Front queue (Frontier Queue)



Back queue (Frontier Queue)



Software Design Principles

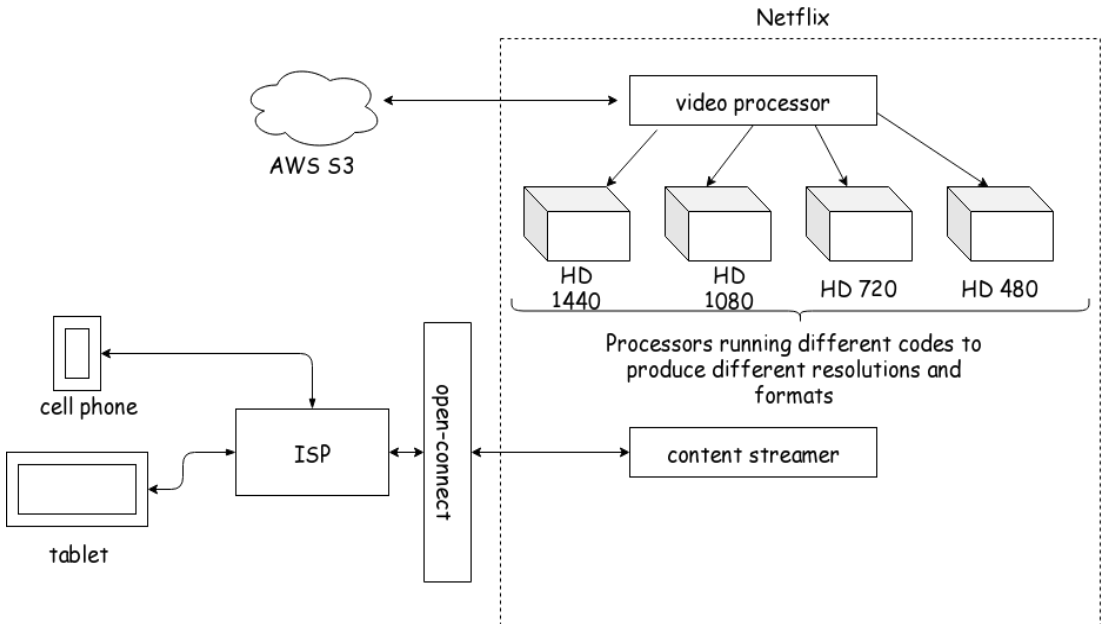
Designing scalable systems

Designing scalable systems

- Adding identical copies of components

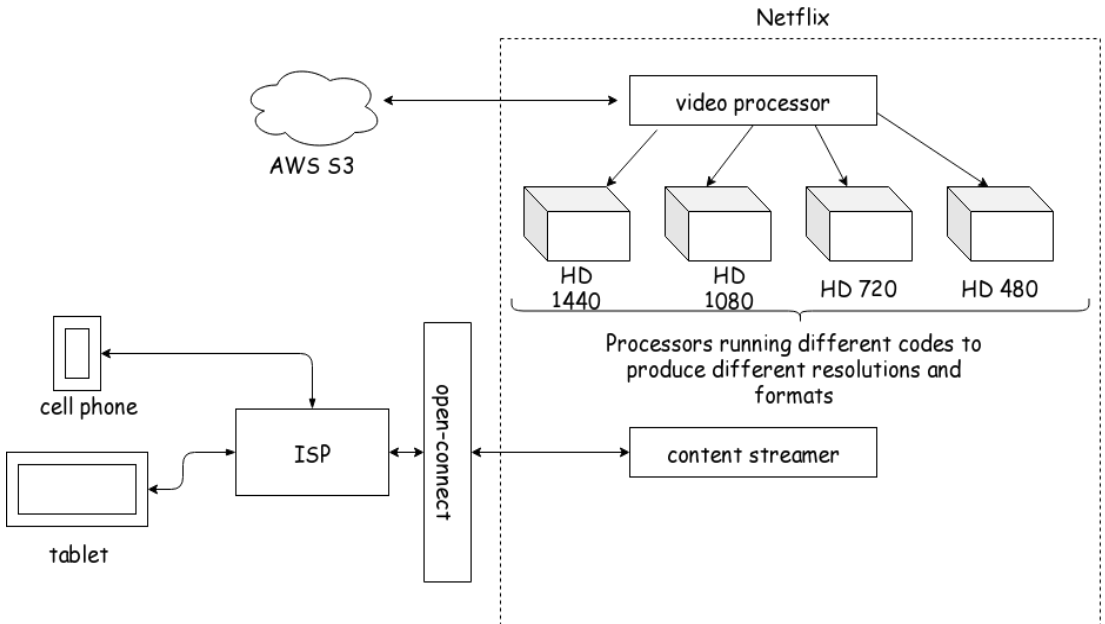
Designing scalable systems

- Adding identical copies of components
- Functional partitioning



Designing scalable systems

- Adding identical copies of components
- Functional partitioning



- Data partitioning

State Management

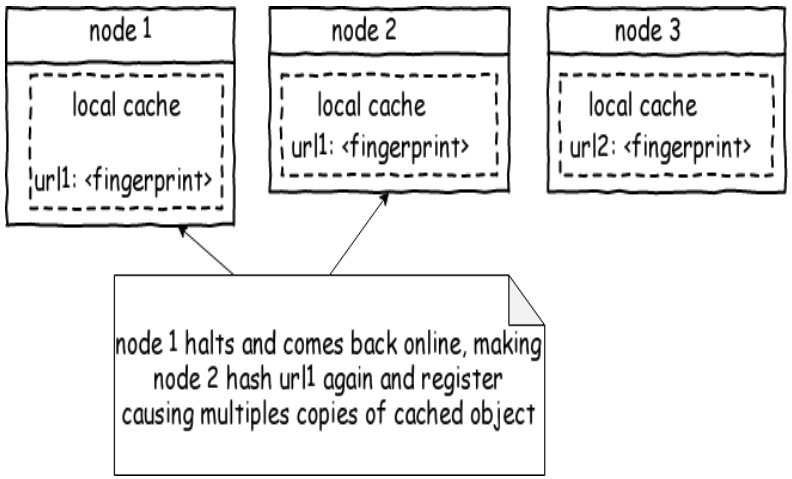


Figure: identical copies of same cached object

State Management

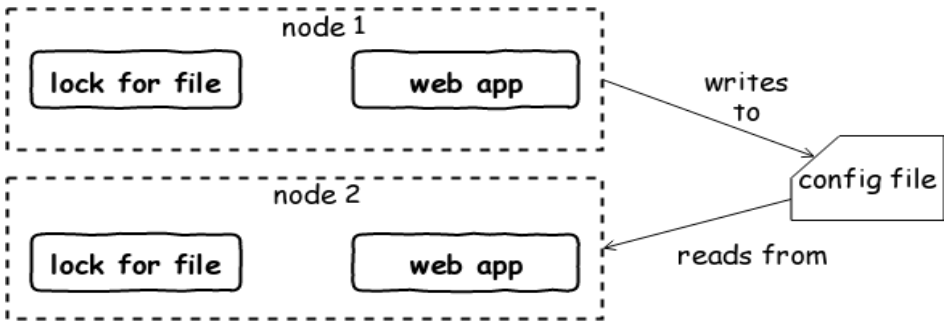


Figure: Using local locks to access shared resources

State Management

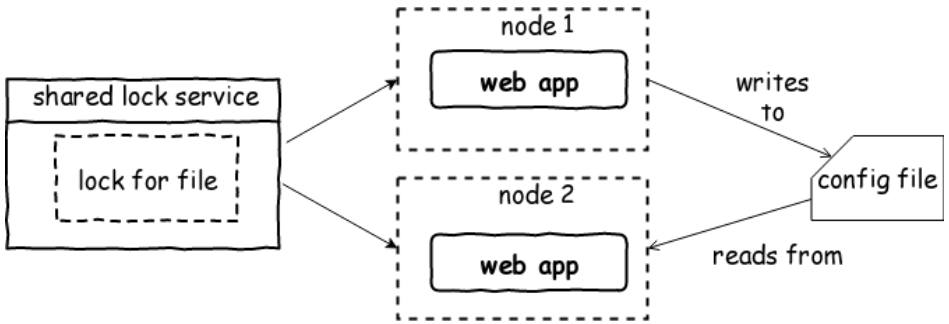
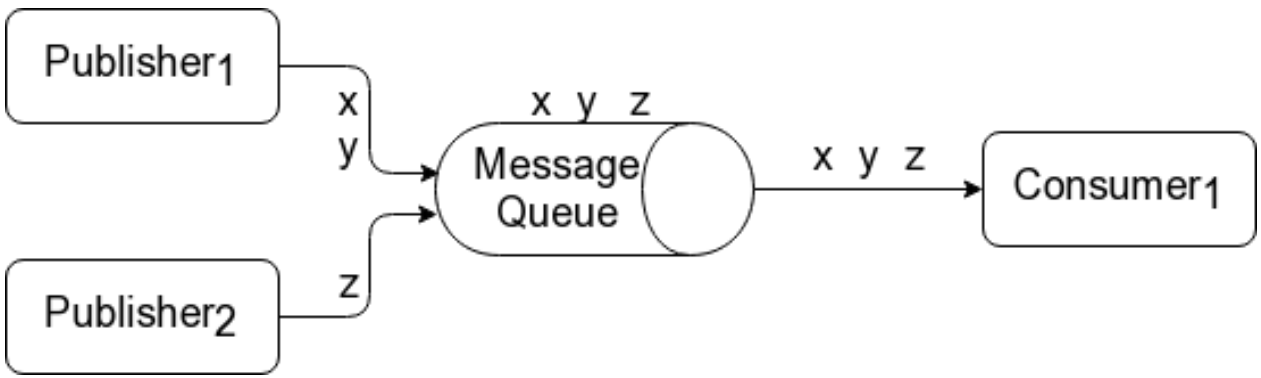


Figure: using shared locks to access shared resources

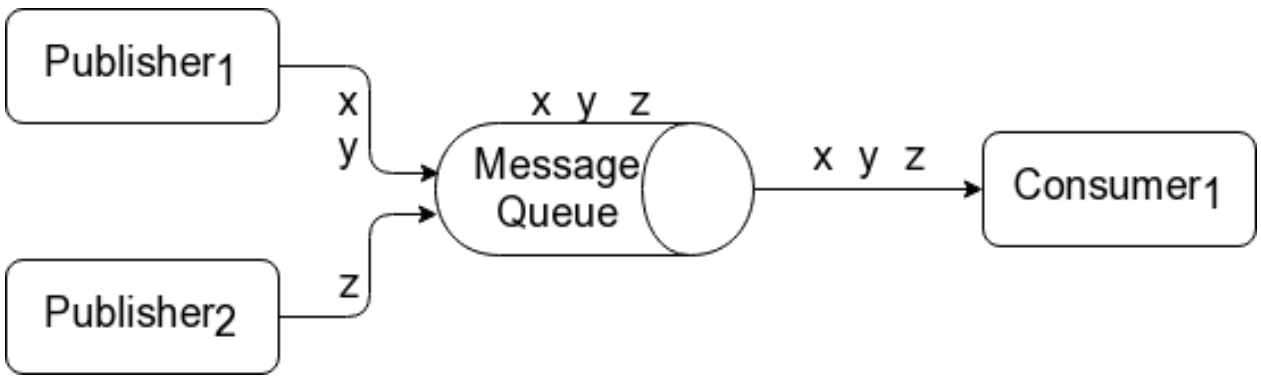
Whirlpool: Event-driven architecture

Message buses

Message Queue(MQ)

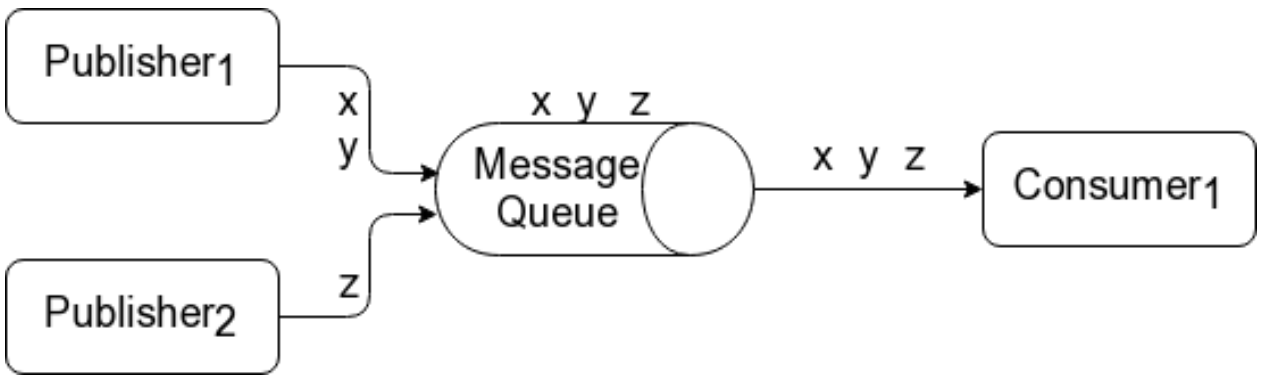


Message Queue(MQ)



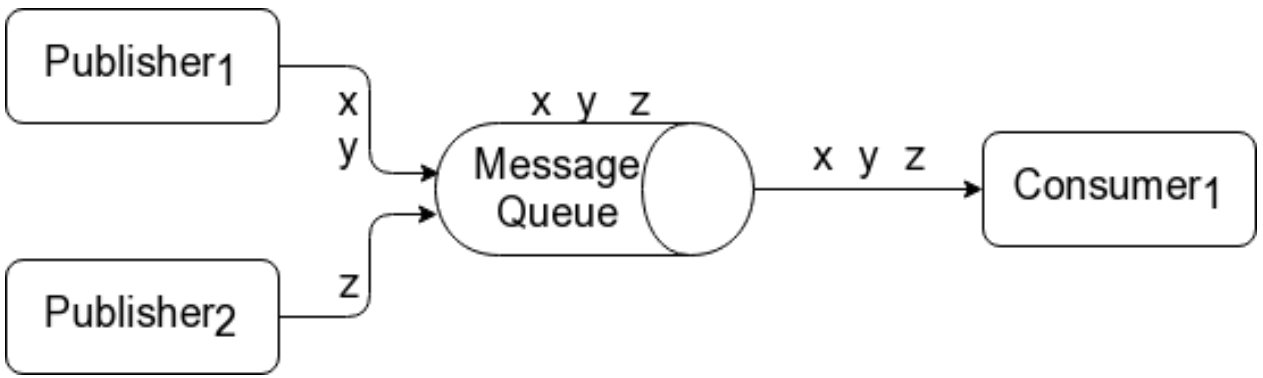
- decoupling

Message Queue(MQ)



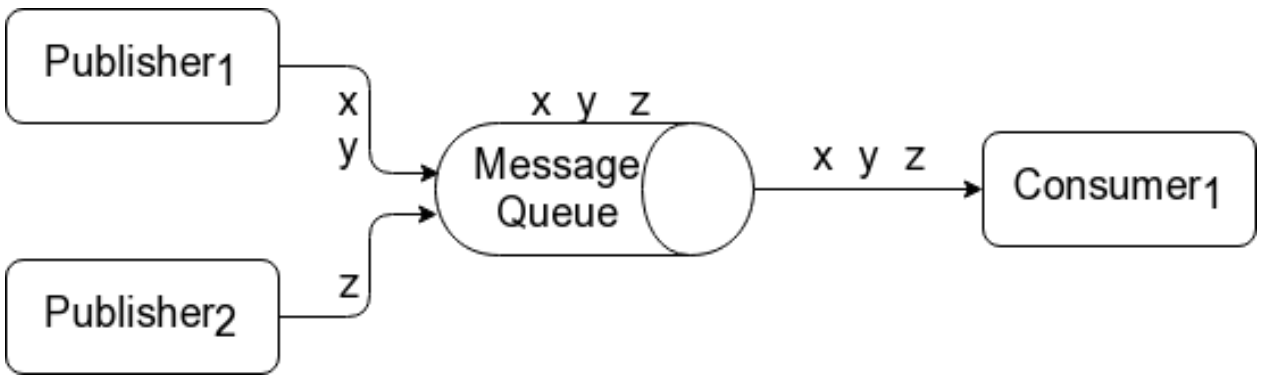
- decoupling
- scaled independently

Message Queue(MQ)



- decoupling
- scaled independently
- balancing traffic

Message Queue(MQ)



- decoupling
- scaled independently
- balancing traffic
- fault-tolerance

MQ: Routing mechanisms

MQ: Routing mechanisms

- Direct Worker Queue Data Flow

MQ: Routing mechanisms

- Direct Worker Queue Data Flow
- Fanout

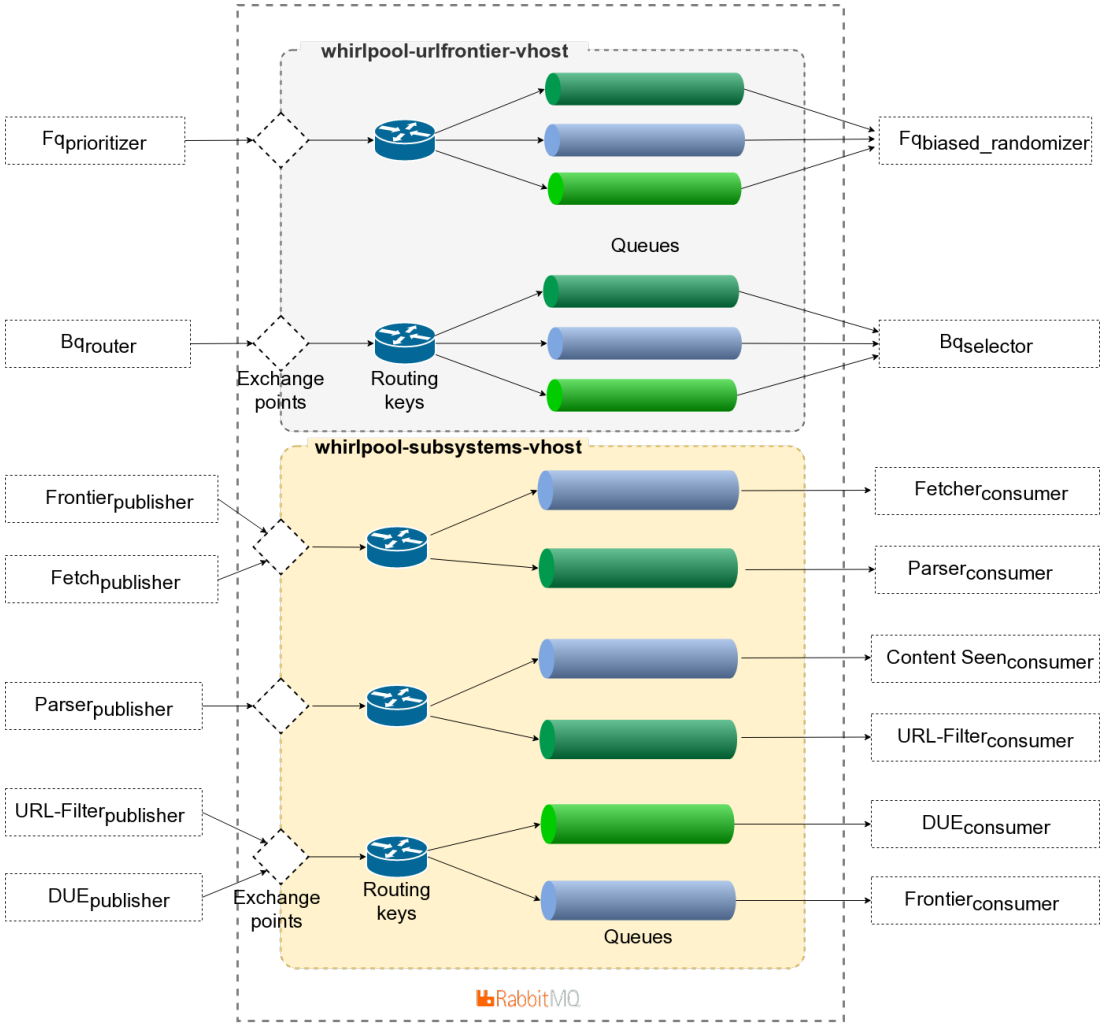
MQ: Routing mechanisms

- Direct Worker Queue Data Flow
- Fanout
- Topic

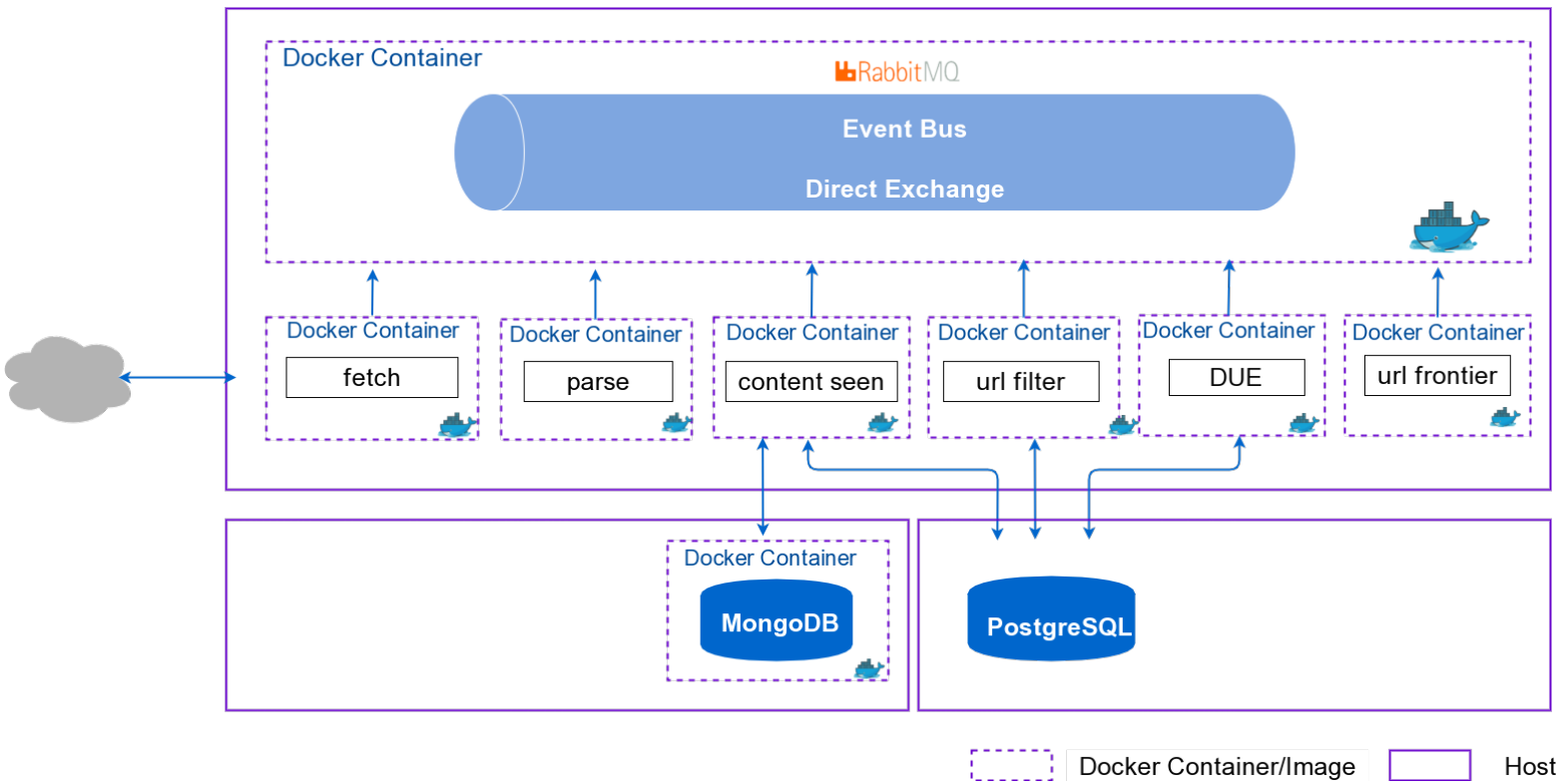
MQ: Routing mechanisms

- Direct Worker Queue Data Flow
- Fanout
- Topic
- Header

Direct Worker Queue Data Flow



RabbitMQ: Message bus



development vs. production docker containers

```yaml
version: '2.4'

networks:
  default:
    external:
      name: whirlpool-net

services:

  base:
    image: python:3.7.4-buster
    command: bash -c "useradd --create-home --shell /bin/bash whirlpool && chown -R whirlpool:whirlpool /home/whirlpool/whirlpool-due"
    volumes:
      - .:/home/whirlpool/whirlpool-due
      - wh-due:/usr/local/lib/python3.7/site-packages
    working_dir: /home/whirlpool/whirlpool-due
    environment:
      - PY_ENV=development
    networks:
      - default

  install:
    extends:
      service: base
    command: pip3 install -r requirements.txt

  quick-up:
    extends:
      service: base
    command: python3 ./due/main.py

volumes:
  wh-due:
    external: true
```

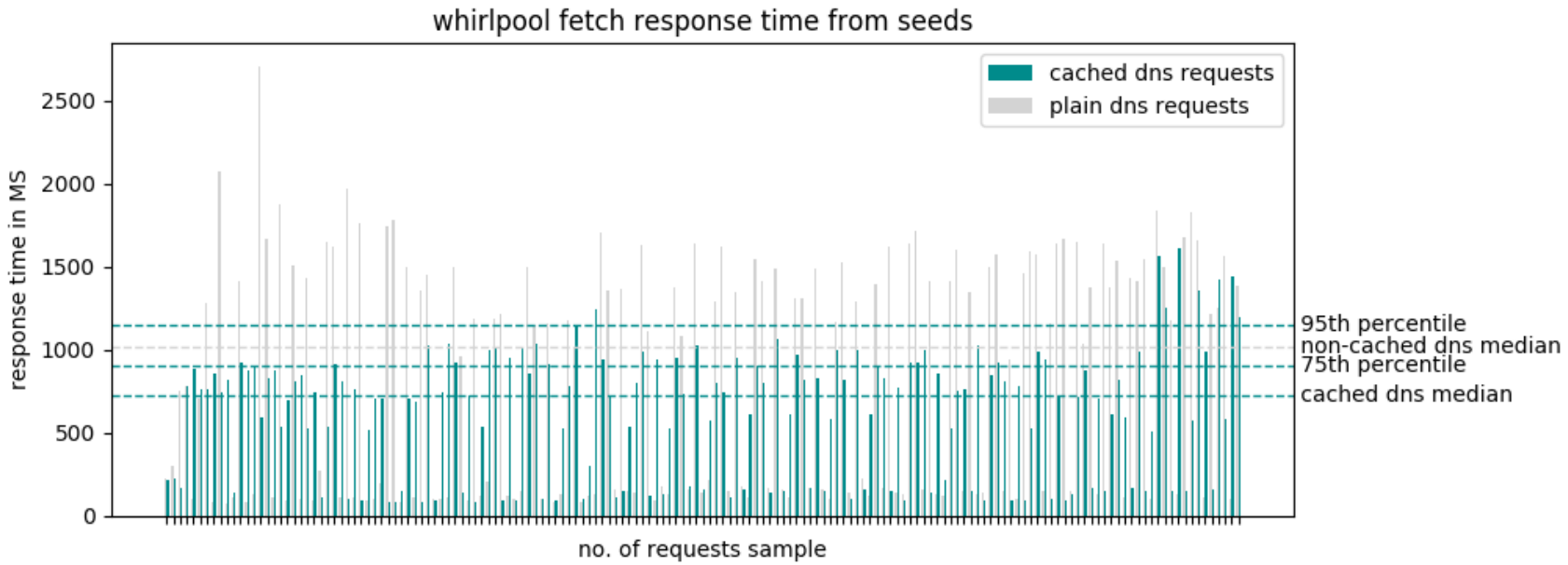
development vs. production docker containers

```
1  FROM python:3.7.4-buster as whirlpool-due-base
2
3  ENV PYTHONDONTWRITEBYTECODE=1
4  ARG WH_DUE_ROOT=/home/whirlpool/whirlpool-due
5  WORKDIR $WH_DUE_ROOT
6
7  RUN apt-get update \
8    && apt-get install -y --no-install-recommends netcat \
9    && rm -rf /var/lib/apt/lists/* \
10   && useradd --create-home --shell /bin/bash whirlpool \
11   && chown -R whirlpool:whirlpool $WH_DUE_ROOT
12
13 # files necessary to build the project
14 COPY .pylintrc ./
15 COPY requirements.txt ./
16
17 RUN mkdir logs/ \
18   && pip3 install -r requirements.txt
19
20 COPY scripts/ scripts/
21 COPY due/ due/
22
23 # docker image for dev target
24 FROM whirlpool-due-base as whirlpool-due-dev
25
26 COPY scripts/wait-for-it.sh scripts/wait-for-it.sh
27 ENTRYPOINT ["bash ./scripts/wait-for-it.sh"]
28
29 # docker image for prod target
30 FROM whirlpool-due-base as whirlpool-due-prod
31
32 COPY scripts/wait-for-it-prod.sh scripts/wait-for-it-prod.sh
33 ENTRYPOINT ["bash ./scripts/wait-for-it-prod.sh"]
```

development vs. production docker containers

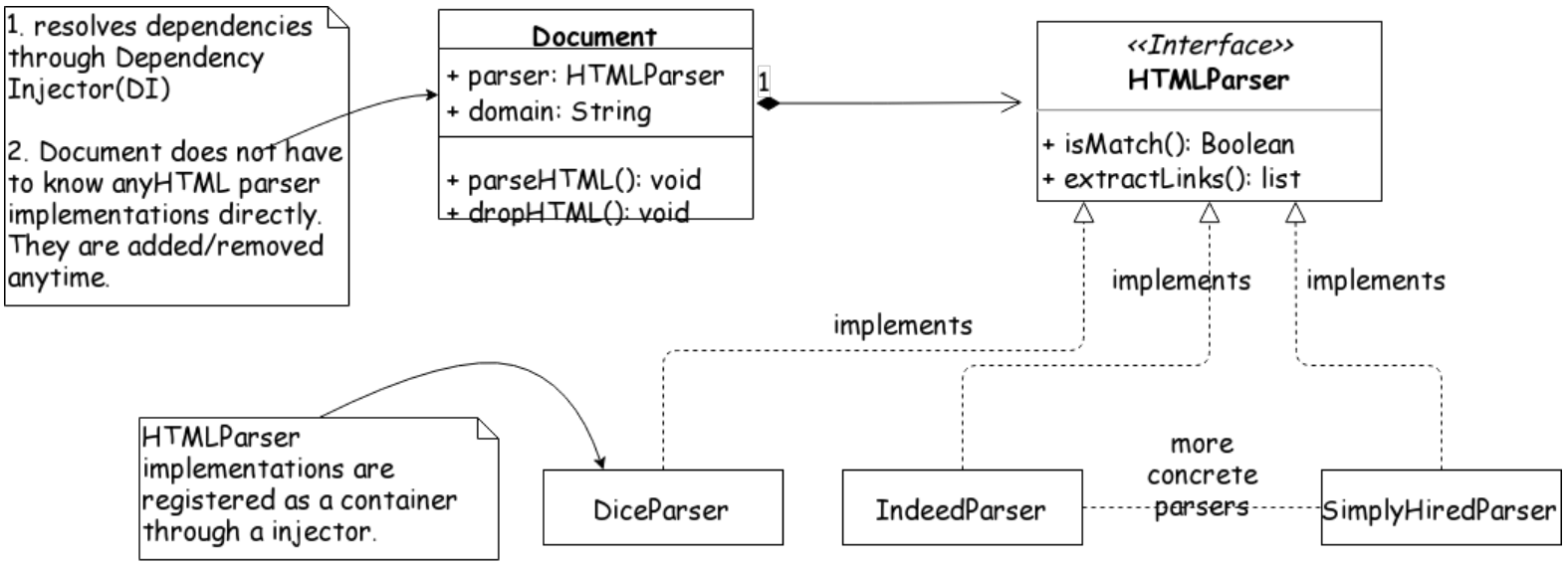
Whirlpool: Fetcher

Whirlpool: Fetcher



Whirlpool: Parser

Parser


1. resolves dependencies through Dependency Injector(DI)
2. Document does not have to know anyHTML parser implementations directly. They are added/removed anytime.
Document
+ parser: HTMLParser
+ domain: String
+ parseHTML(): void
+ dropHTML(): void
1
<<Interface>>
HTMLParser
+ isMatch(): Boolean
+ extractLinks(): list
implements
implements
implements
HTMLParser implementations are registered as a container through a injector.
DiceParser
IndeedParser
more concrete parsers
SimplyHiredParser

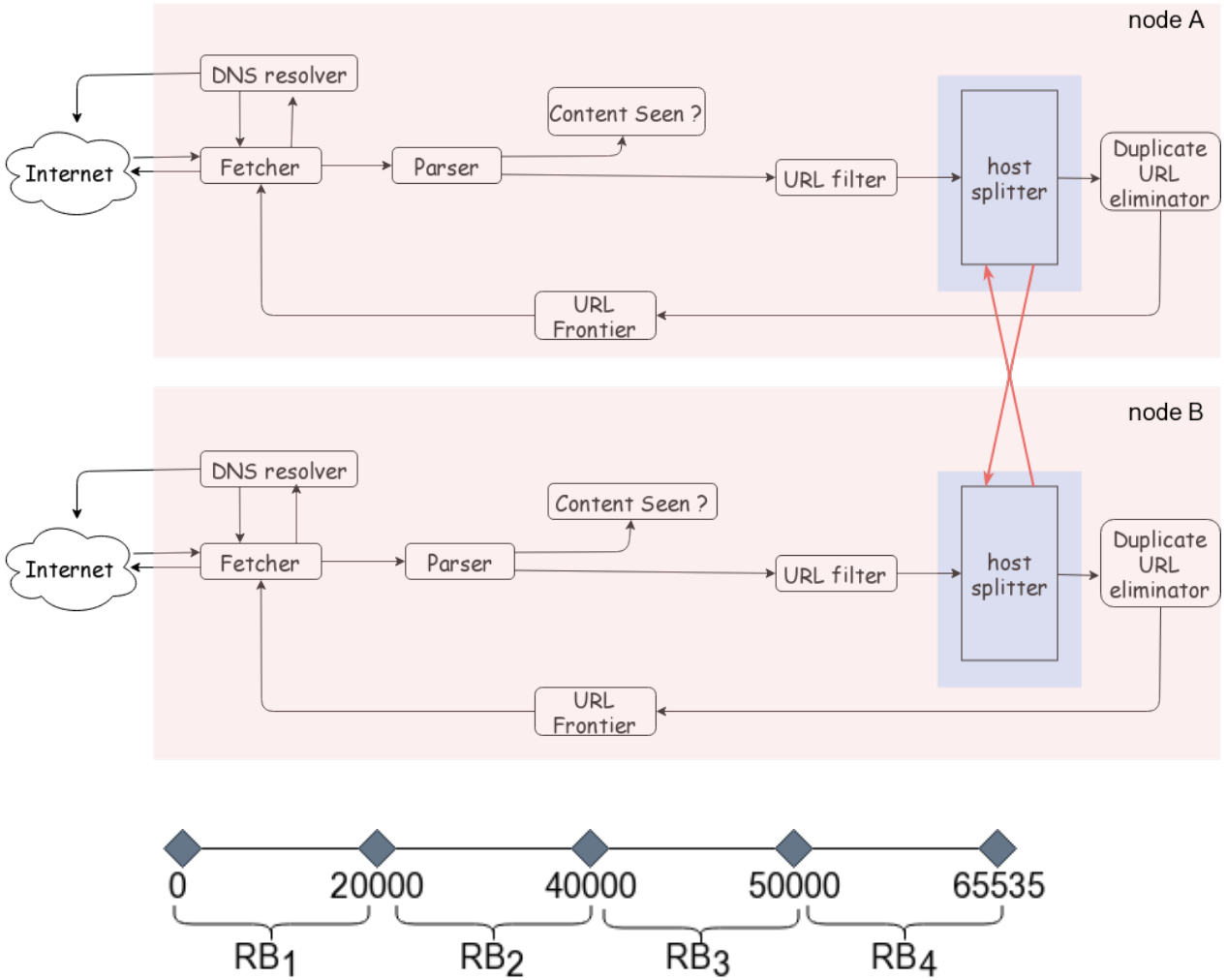
Whirlpool: Deduplication

Deduplication

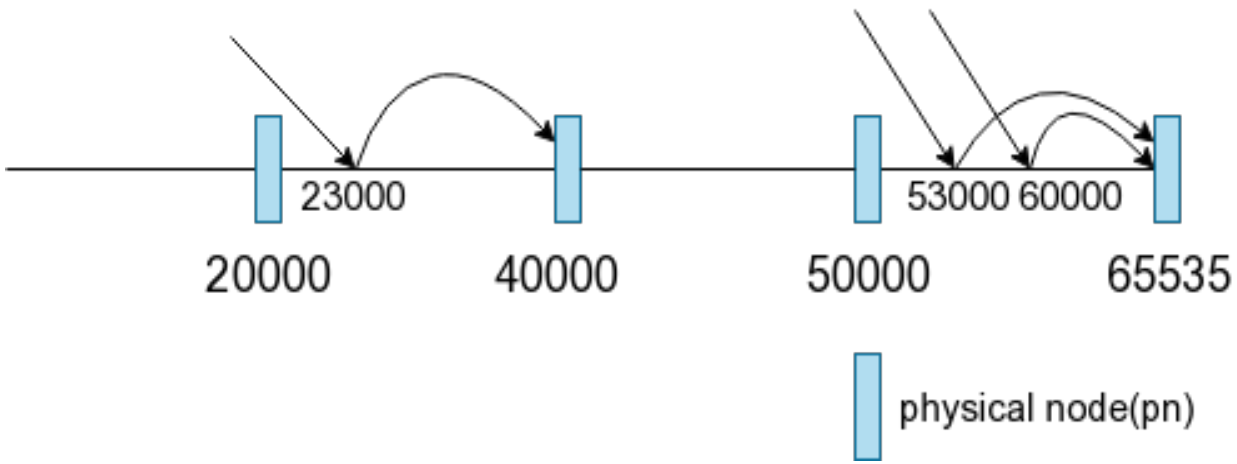
to add something

Whirlpool: Distributed Crawling

Host splitting in Mercator



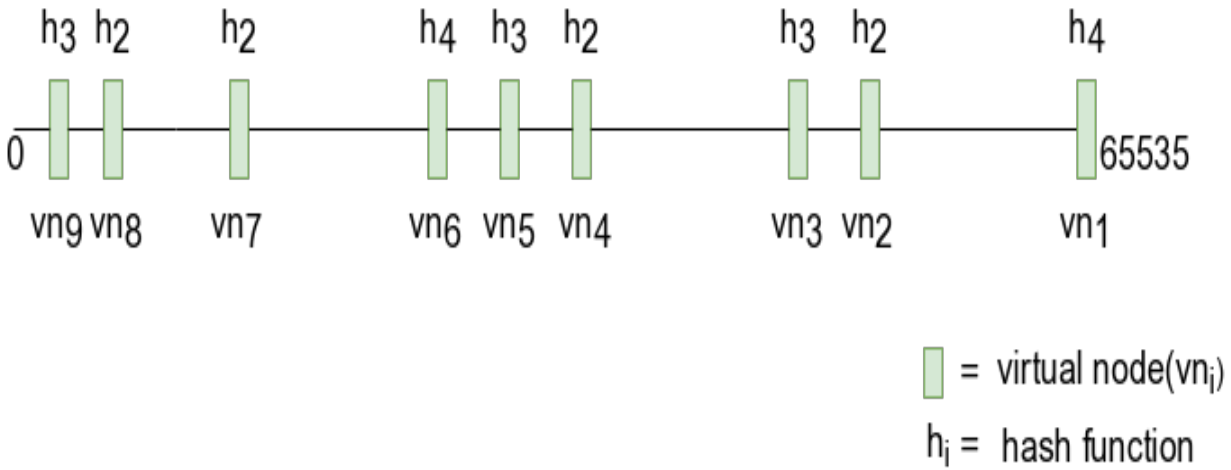
Rebalancing: mod N



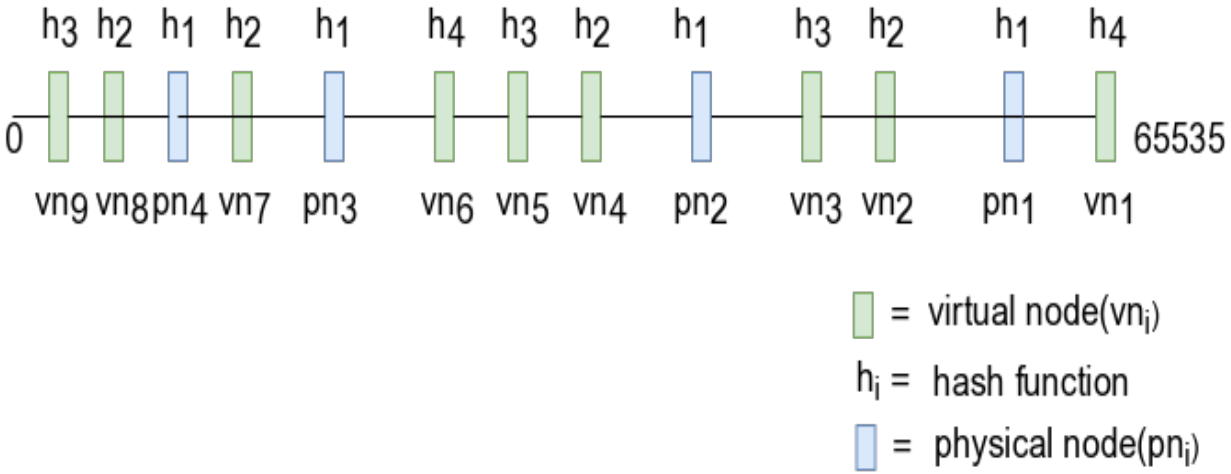
Rebalancing: mod N

absolute URL	m = hash(absolute URL)	m mod N(1)	m mod N(2)	m mod N(3)	m mod N(4)
jobs.com/browses	24	n1	n1 0	n1 0	n1 0
dice.com/browse-titles	59956	n1	n2 1	n1 1	n4 1
monster.com/categories	10090	n1	n1 0	n1 0	n3 1
simplyhired.com/find-jobs	23000	n1	n2 1	n2 0	n4 1
indeed.com/jobs/unix-admin	41103	n1	n2 1	n1 1	n4 1
glassdoor.com/hires/software-developer	35114	n1	n1 0	n3 1	n3 0
data movement		0%	50%	50%	60%

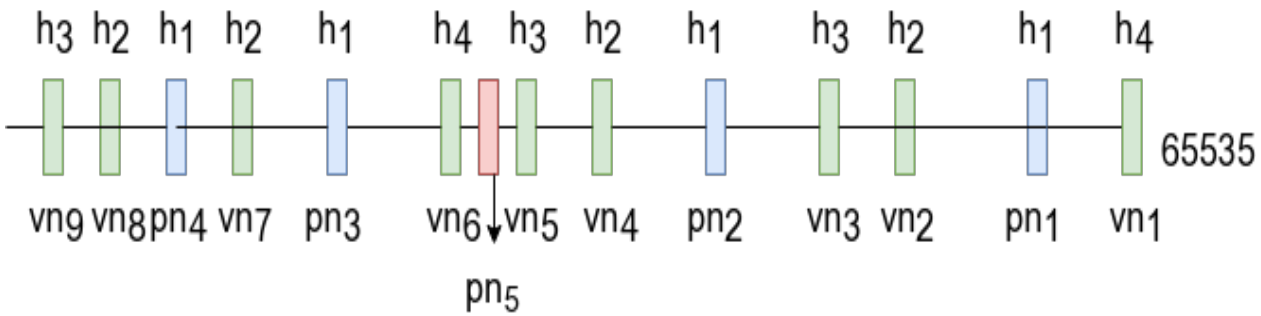
Rebalancing: virtual nodes



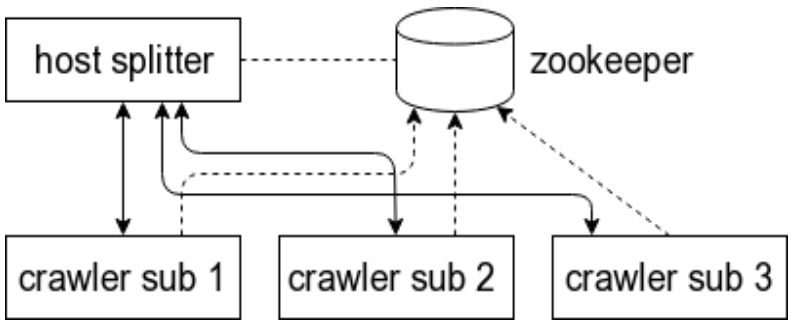
Rebalancing: virtual nodes



Rebalancing: virtual nodes



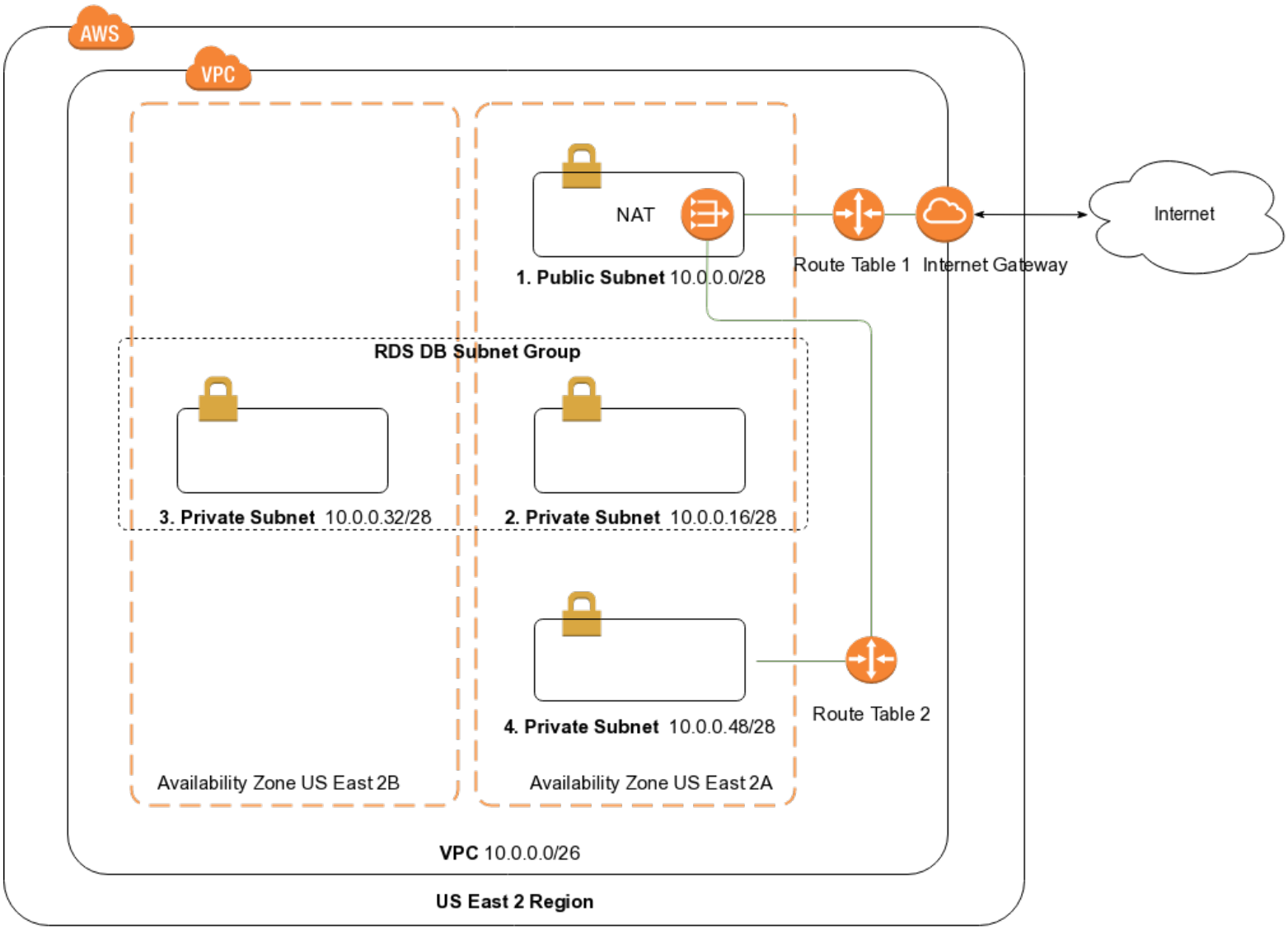
Zookeeper



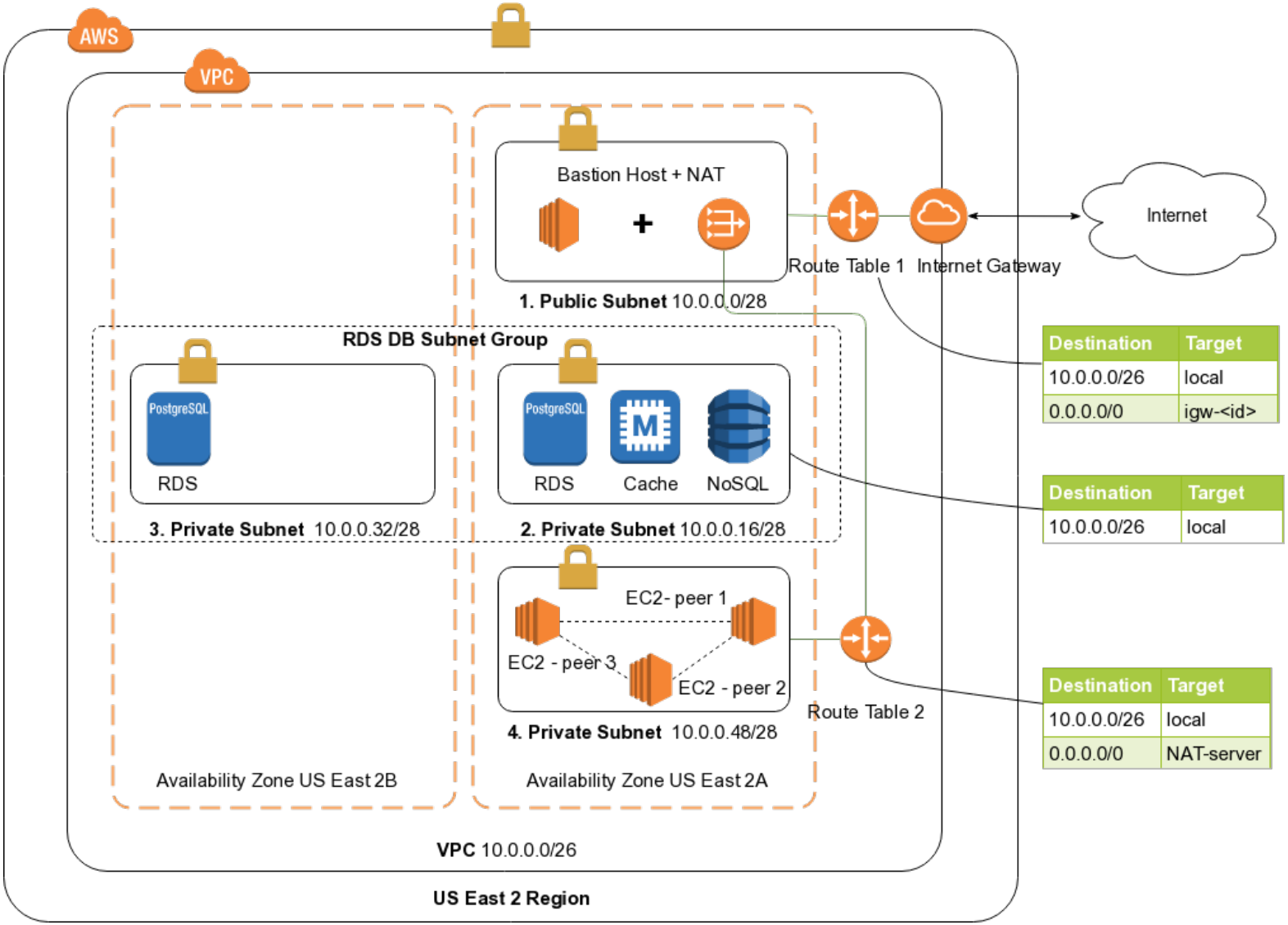
hash range	vnode(vni)	pnode(pni)	IP
0 – 3k	vn9	pn4	10.0.0.4
3k – 6k	vn8	pn4	10.0.0.4
6k – 10k	vn7	pn3	10.0.0.3
10k – 15k	vn6	pn2	10.0.0.2
15k – 20k	vn5	pn2	10.0.0.2
20k – 35k	vn4	pn2	10.0.0.2
35k – 45k	vn3	pn1	10.0.0.1
45k – 55k	vn2	pn1	10.0.0.1
55k – 65k	vn1	pn4	10.0.0.4

Whirlpool: Operations

From 10,000 ft.



From 5,000 ft.


AWS
VPC
Bastion Host + NAT
1. Public Subnet 10.0.0.0/28
Route Table 1
Internet Gateway
Internet
RDS DB Subnet Group
PostgreSQL
RDS
3. Private Subnet 10.0.0.32/28
PostgreSQL
RDS
Cache
NoSQL
2. Private Subnet 10.0.0.16/28
EC2- peer 1
EC2 - peer 3
EC2 - peer 2
4. Private Subnet 10.0.0.48/28
Route Table 2
Availability Zone US East 2B
Availability Zone US East 2A
VPC 10.0.0.0/26
US East 2 Region
Destination Target
10.0.0.0/26 local
0.0.0.0/0 igw-<id>
Destination Target
10.0.0.0/26 local
Destination Target
10.0.0.0/26 local
0.0.0.0/0 NAT-server

Infrastructure as a code

Future work

future to do

to add something

Thank you! Questions ?