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 27, 2019

Channel Islands
CALIFORNIA STATE UNIVERSITY

- 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

$\left.\begin{array}{l}\text{Crawler characteristics \& history}\\\text{Mercator 1999 (Heydon \& Najork)}\\\text{Software Design Principles}\end{array}\right\}$ background

- Motivation & Contributions

$$\left.\begin{array}{l}\text{• Crawler characteristics \& history}\\ \text{• Mercator 1999 (Heydon \& Najork)}\\ \text{• Software Design Principles}\end{array}\right\}\ \text{background}$$

- Whirlpool: Event driven architecture

- Motivation & Contributions
- Crawler characteristics & history
- Mercator 1999 (Heydon & Najork)
- Software Design Principles
- Whirlpool: Event driven architecture
- Whirlpool: Fetcher

background

- 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

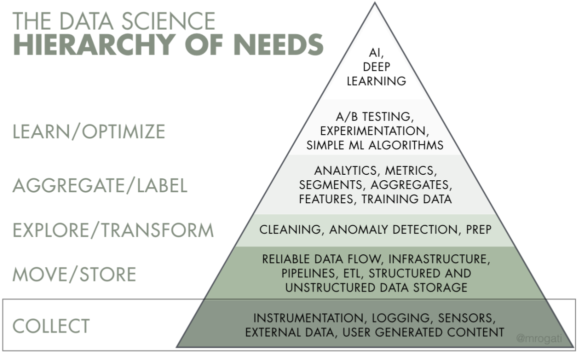
- Whirlpool: Event driven architecture
- Whirlpool: Fetcher
- Whirlpool: Parser
- Whirlpool: Deduplication
- Whirlpool: Distributed Crawling
- Whirlpool: Operations

implementation & results

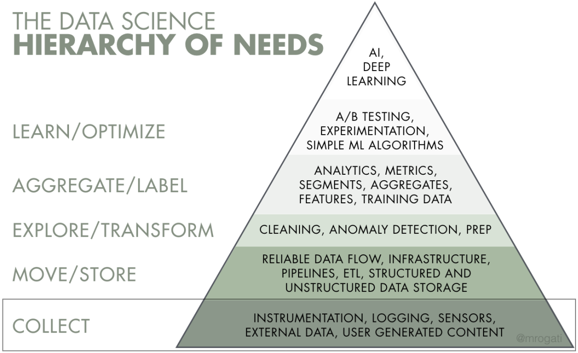
- Future work

Motivation & Contribution

Motivation


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

Motivation


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

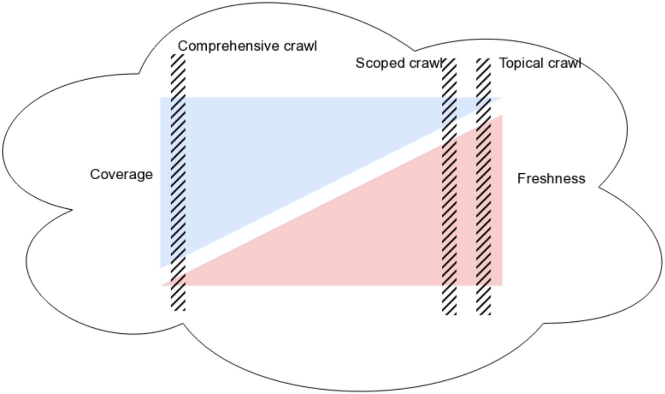
Self-actualization (AI) is great, but you first need food, water, and shelter (data literacy, collection, and infrastructure)."

Contributions

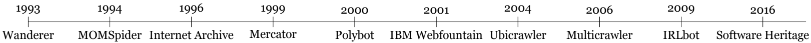
to be completed

Crawler characteristics & history

Coverage & Freshness



Web crawlers (1990 - 2019)



1993	1994	1996	1999	2000	2001	2004	2006	2009	2016
Wanderer	MOMSpider	Internet Archive	Mercator	Polybot	IBM Webfountain	Ubicrawler	Multicrawler	IRLbot	Software Heritage

Mercator 1999 (Heydon & Najork)

basic crawling algorithm

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

Mercator background

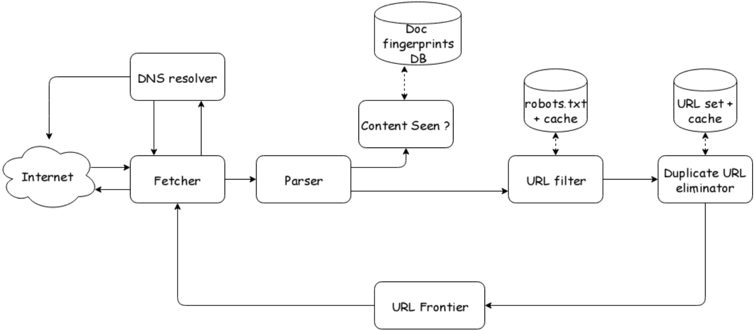
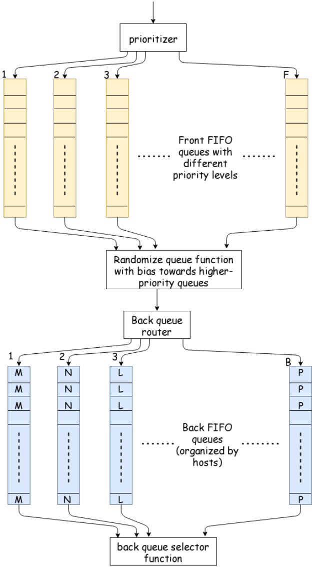
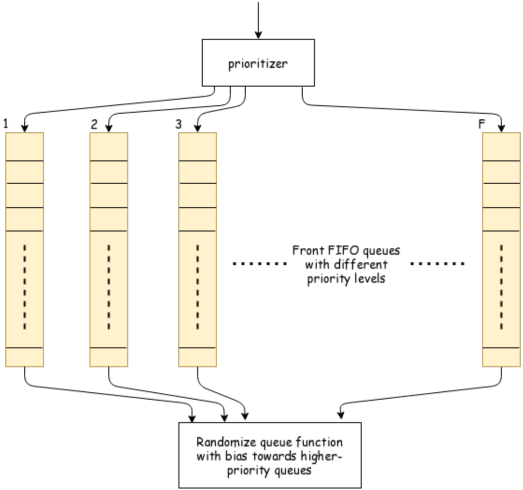


Figure: Mercator building blocks (Heydon & Najork)

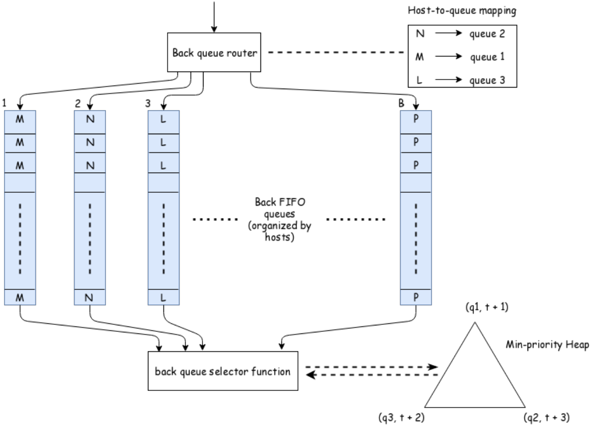
URL Frontier Scheme


prioritizer
1
2
3
F
Front FIFO queues with different priority levels
Randomize queue function with bias towards higher-priority queues
Back queue router
1
2
3
B
M
N
L
P
Back FIFO queues (organized by hosts)
back queue selector function

Front queue (Frontier Queue)


prioritizer
1
2
3
F
Front FIFO queues with different priority levels
Randomize queue function with bias towards higher-priority queues

Back queue (Frontier Queue)



Software Design Principles

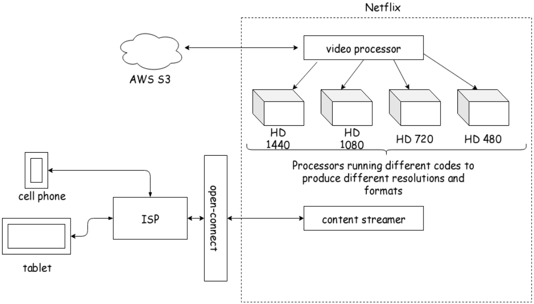
Designing scalable systems

Designing scalable systems

- Adding identical copies of components

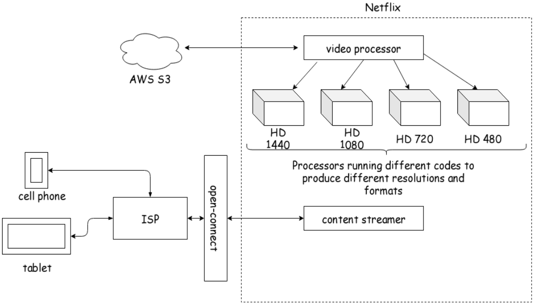
Designing scalable systems

- Adding identical copies of components
- Functional partitioning



Designing scalable systems

- Adding identical copies of components
- Functional partitioning



- Data partitioning

State Management

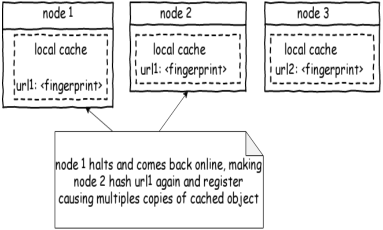

node 1
local cache
url1: <fingerprint>
node 2
local cache
url1: <fingerprint>
node 3
local cache
url2: <fingerprint>
node 1 halts and comes back online, making
node 2 hash url1 again and register
causing multiples copies of cached object

Figure: identical copies of same cached object

State Management

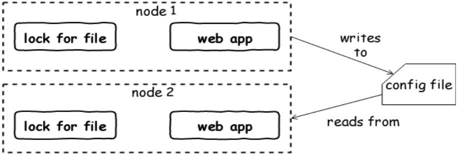


Figure: Using local locks to access shared resources

State Management

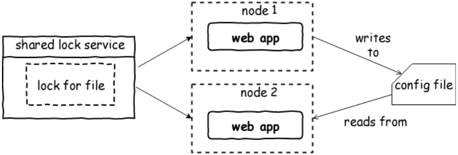
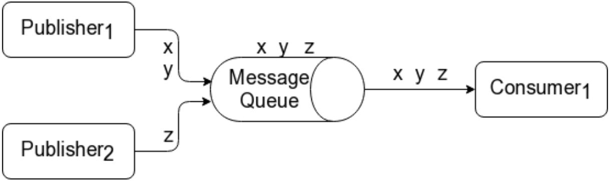


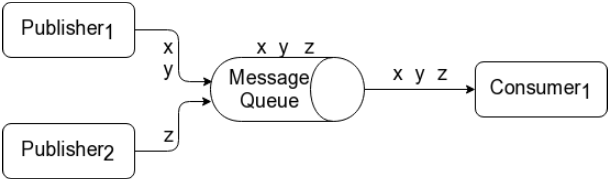
Figure: using shared locks to access shared resources

Whirlpool: Event-driven architecture

Message Queue(MQ)

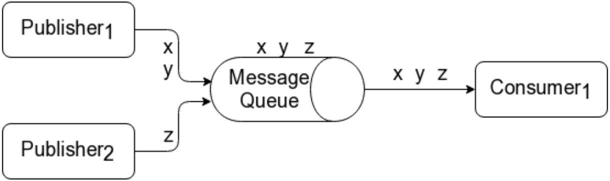


Message Queue(MQ)



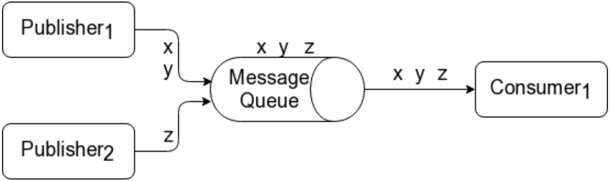
- decoupling

Message Queue(MQ)



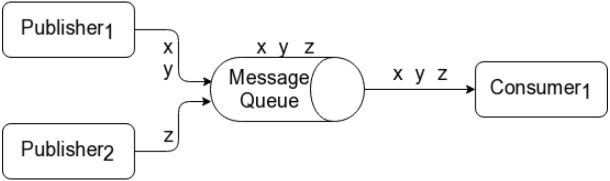
- decoupling
- scaled independently

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

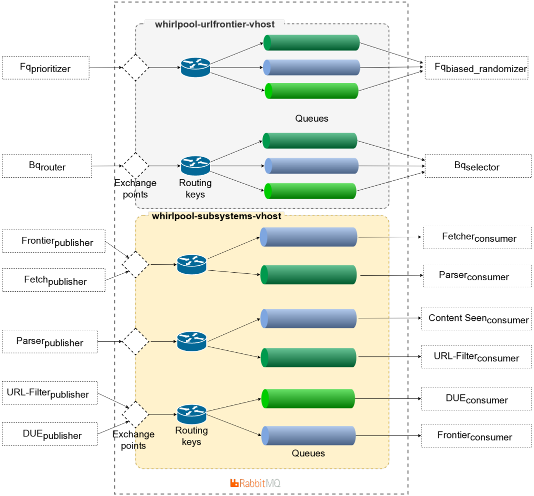
MQ: Routing mechanisms

- Direct Worker Queue Data Flow
- Fanout
- Topic

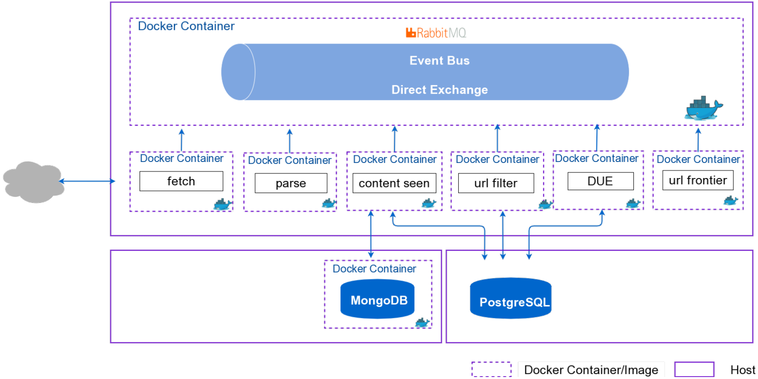
MQ: Routing mechanisms

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

Direct Worker Queue Data Flow



RabbitMQ: Message bus



development vs. production docker containers

things to add

Whirlpool: Fetcher

Whirlpool: Parser

Parser

to add something

Whirlpool: Near-Deduplication

Dedupe

to add something

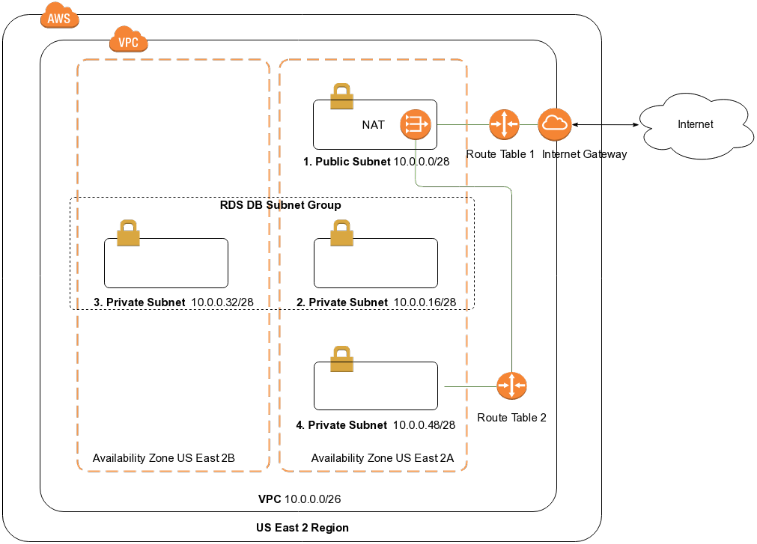
Whirlpool: Distributed Crawling

Dist. crawl

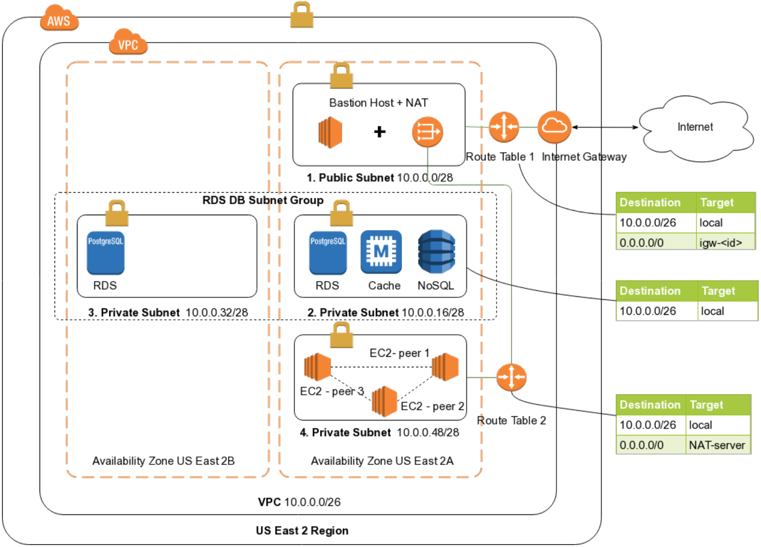
to add something

Whirlpool: Operations

From 10,000 ft.


AWS
VPC
NAT
1. Public Subnet 10.0.0.0/28
Route Table 1
Internet Gateway
Internet
RDS DB Subnet Group
3. Private Subnet 10.0.0.32/28
2. Private Subnet 10.0.0.16/28
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

From 5,000 ft.



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

Future work

future to do

to add something

Thank you! Questions ?