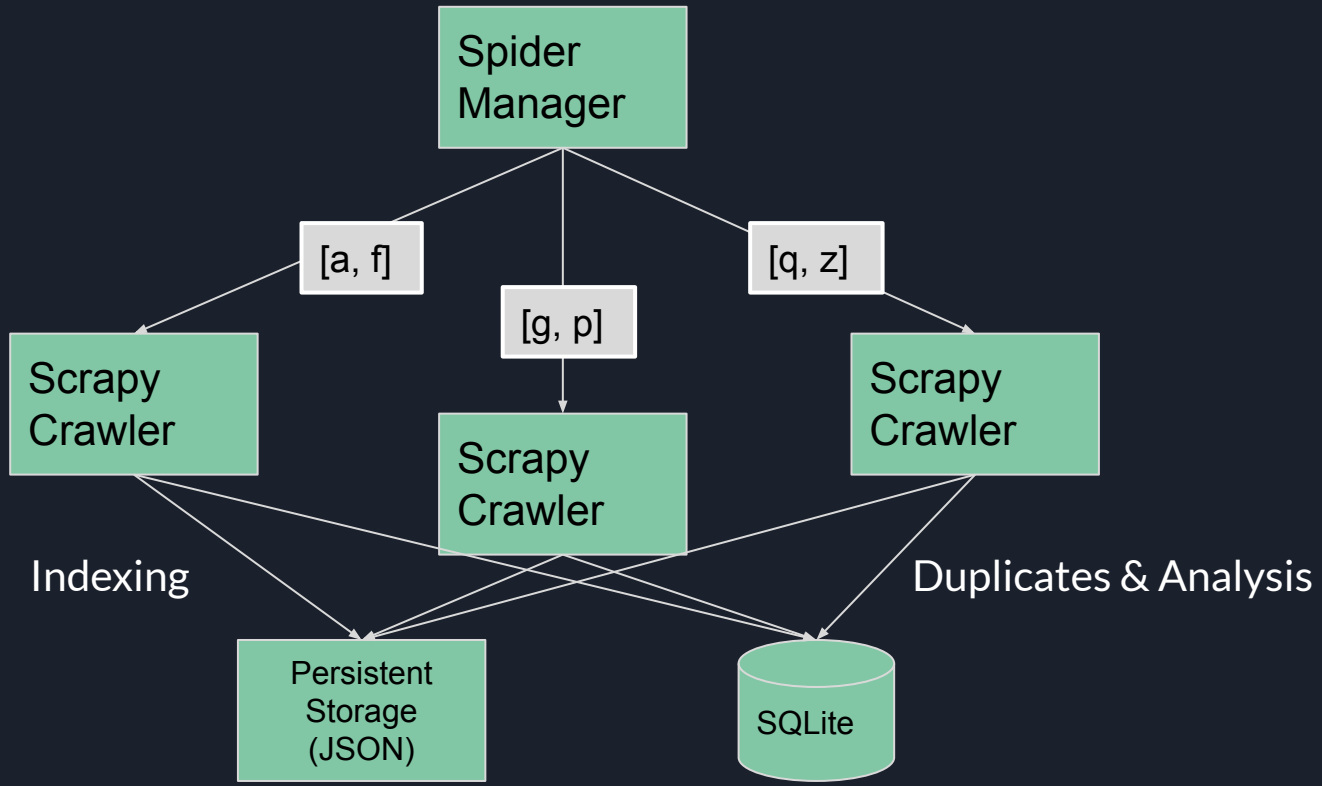
Group 10 Demo

Cy Heffley, Connor Cole, Eduardo Rodriguez, Zach McGee

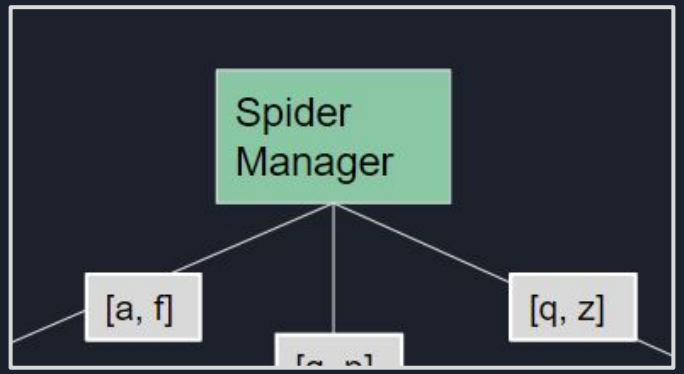
System Architecture



Spider Manager

Purpose: Divide up different suffixes across multiple scrapers

In true distributed system: manager would be responsible for communication with neighbors, rebalancing suffix rules/ conditions



Code Walkthrough #1: Table Creation

```
table_name =
        "subprocesses_{}_initialdepth_{}_filterdepth_{}_maxdepth_{}".format(subp
        rocesses, initial_depth, filter_depth, max_depth)
conn = sqlite3.connect(".\\db\\urldatabase.db")
print(sqlite3.version)
cur = conn.cursor()


try:
    cur.execute("CREATE TABLE IF NOT EXISTS url (url_id INTEGER NOT NULL
            PRIMARY KEY AUTOINCREMENT, scraper_id INT NOT NULL, url
            VARCHAR(255) NOT NULL, creation_date TIMESTAMP DEFAULT
            CURRENT_TIMESTAMP)")
    conn.commit()
except Error as e:
    print(e)
    time.sleep(4)

#Clear out table of URLs for new scrapers to write to
try:
    cur.execute("delete from url")
except:
    pass
```

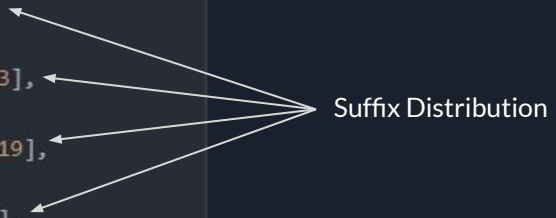
```
#Create fresh table to track stats with
cur.execute("DROP TABLE IF EXISTS {}".format(table_name))
cur.execute('''
    CREATE TABLE {} (
    insertion_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    scraper_name VARCHAR(100),
    start_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    enqueued INT,
    dequeued INT,
    request_count INT,
    response_count INT,
    duplicates_filtered_internal INT,
    duplicates_filtered_from_others INT,
    offsites_filtered INT,
    computing_count INT,
    undergrad_count INT,
    research_count INT,
    online_count INT,
    current_time TIMESTAMP,
    subprocesses INT DEFAULT {},
    initial_depth INT DEFAULT {},
    filter_depth INT DEFAULT {},
    max_depth INT DEFAULT {}
    )

    '''.format(table_name, subprocesses, initial_depth, filter_depth,
            max_depth))
conn.commit()
conn.close()
```

Code Walkthrough #2: Scraper Creation

```
import string
alphabet_list = list(string.ascii_lowercase)

if subprocesses == 1:
    runner.crawl(SuperSpider, allowed_suffix= alphabet_list, name="scraper_1",
            table_name = table_name)
elif subprocesses == 2:
    runner.crawl(SuperSpider, allowed_suffix= alphabet_list[:13],
            name="scraper_1", table_name = table_name)
    runner.crawl(SuperSpider, allowed_suffix = alphabet_list[13:],
            name="scraper_2",  table_name = table_name)
elif subprocesses == 4:
    runner.crawl(SuperSpider, allowed_suffix= alphabet_list[:6],
            name="scraper_1", table_name = table_name)
    runner.crawl(SuperSpider, allowed_suffix = alphabet_list[6:13],
            name="scraper_2",  table_name = table_name)
    runner.crawl(SuperSpider, allowed_suffix = alphabet_list[13:19],
            name="scraper_3",  table_name = table_name)
    runner.crawl(SuperSpider, allowed_suffix = alphabet_list[19:],
            name="scraper_4",  table_name = table_name)
```



Code Walkthrough #3: Scraper Logic

```
def parse(self, response):
    self.log("scraper {}".format(self.name))
    self.log('crawling {}'.format(response.url))
    self.log('current depth: {}'.format(response.meta['depth']))

    already_in_db = self.check_in_db(response.url)
    suffix = self.get_suffix

    #If we already have the link in DB, only consider it if its in the
            inital sweep
    if already_in_db:
        self.duplicates_from_other_scraper += 1
        if response.meta['depth'] > self.initial_depth:
            return

    #Periodically add stats to DB
    if self.crawler.stats.get_stats()["scheduler/dequeued"] -
            self.last_dequeue_value > 500:
        self.last_dequeue_value = self.crawler.stats.get_stats()["scheduler/
                dequeued"]
        self.write_stats()

    #Only add keyword info if this is a new page- else, just extract links!
    if not already_in_db:
        for keyword in self.keywords:
            if keyword in response.text.lower():
                self.keywords[keyword].append(response.url)
        self.add_to_db(response.url)

    #This part adds any new links we want to consider
    for link in self.link_extractor.extract_links(response):
        suffix = self.get_suffix(link)
        if (response.meta['depth'] > self.filter_depth) and (suffix not in
                self.suffix_list):
            print("Filtered out non matching suffix!")
        else:
            yield scrapy.Request(link.url, callback=self.parse)
```

} Database check from other scrapers

} Add stats to database for analysis

} Add keyword info if new page

} Extract links, check filter depth and suffix, and enqueue pages

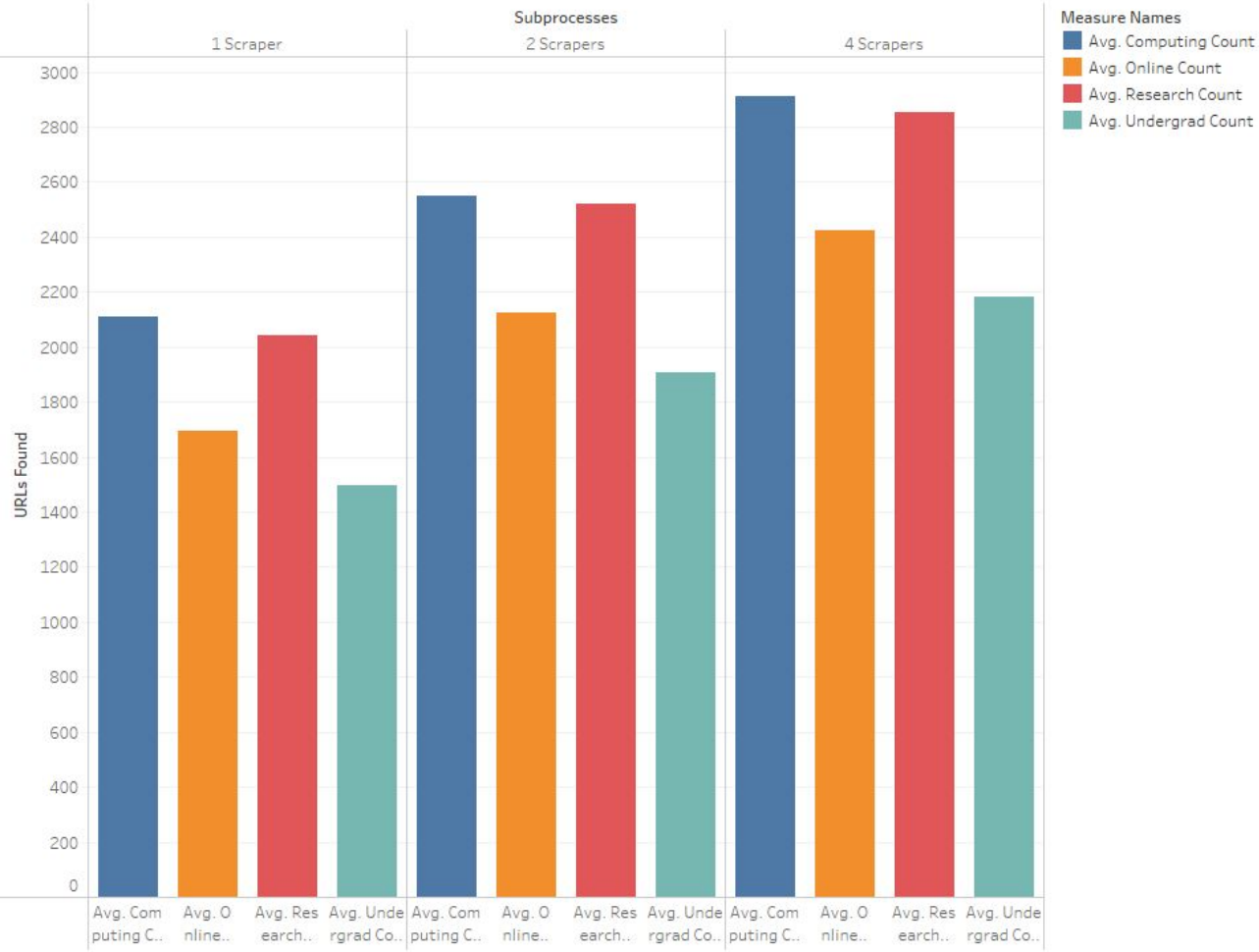
Crawling Demonstration

Crawling Results of Our Spider Configurations

Evaluation methods/metrics/datasets and results

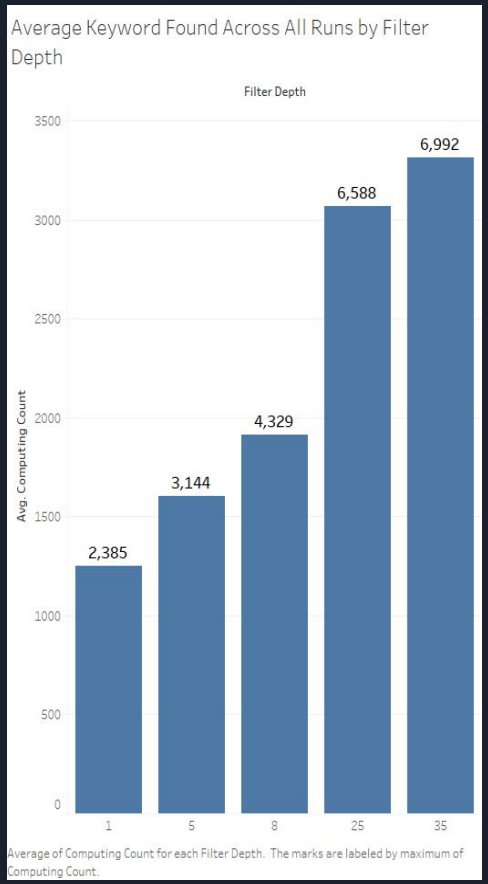
- Compare subprocesses, initial, filter, max depths on:
  - Keywords found
  - Duplicate URLs filtered over time
  - Offsite URLs filtered over time
  - GET Requests made over time

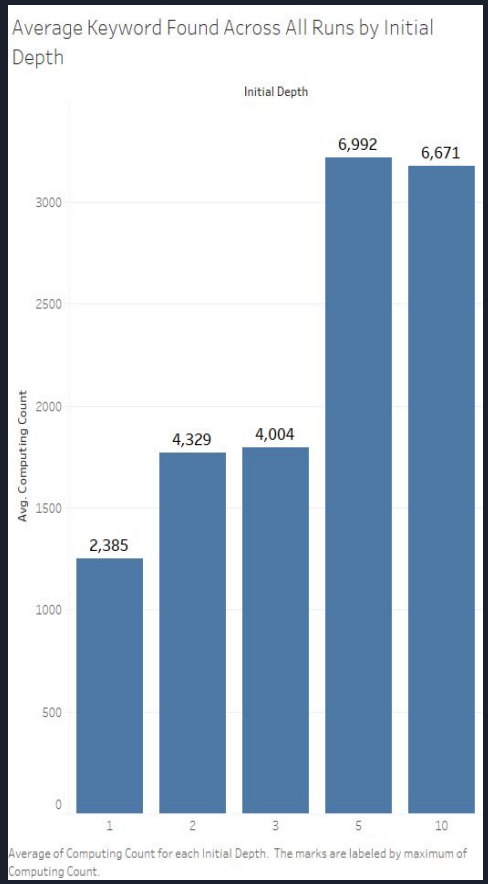
Average Keywords Found By Subprocesses

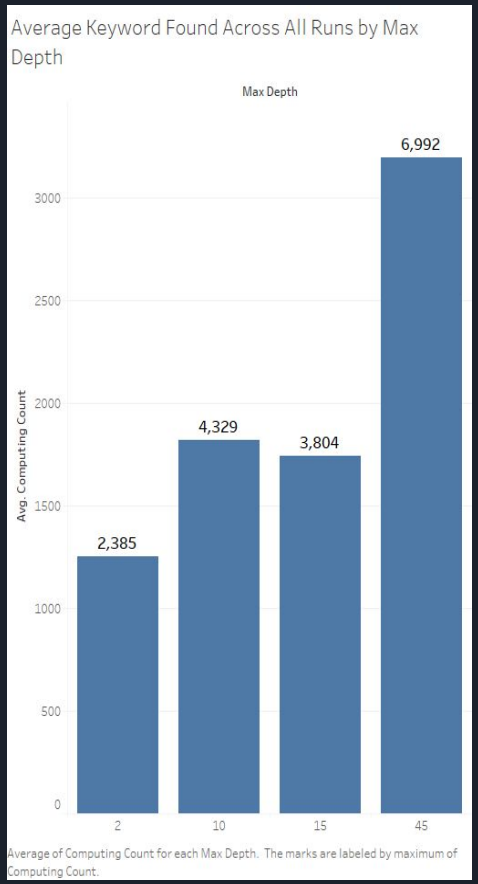


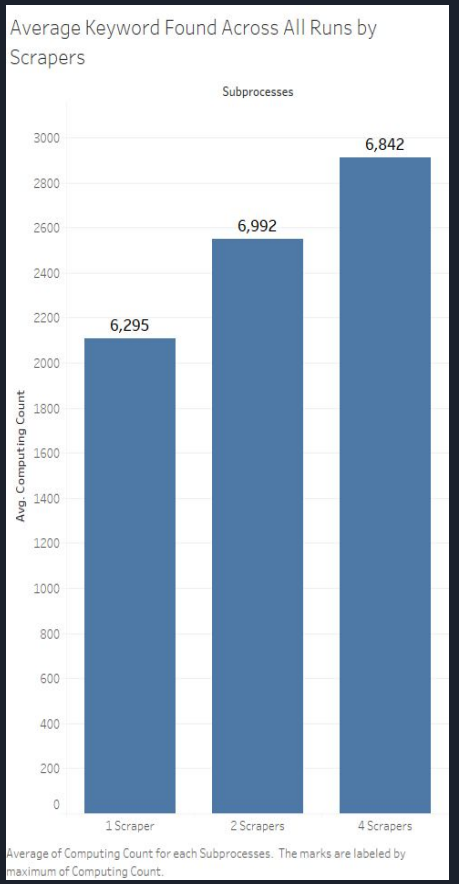
Avg. Computing Count, Avg. Online Count, Avg. Research Count and Avg. Undergrad Count for each Subprocesses. Color shows details about Avg. Computing Count, Avg. Online Count, Avg. Research Count and Avg. Undergrad Count.

We see that generally across all run configurations, adding more scrapers gives better results for each keyword.





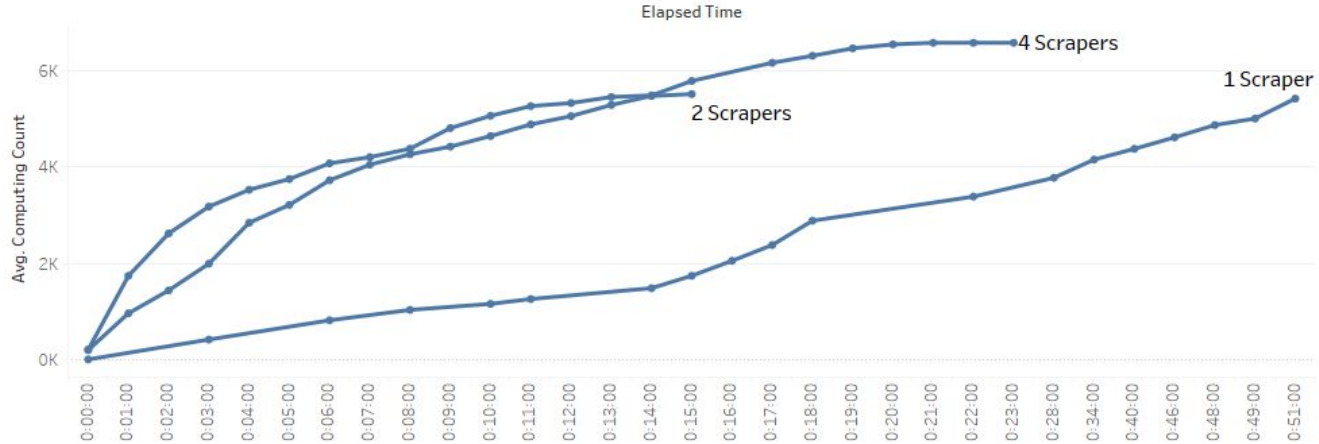




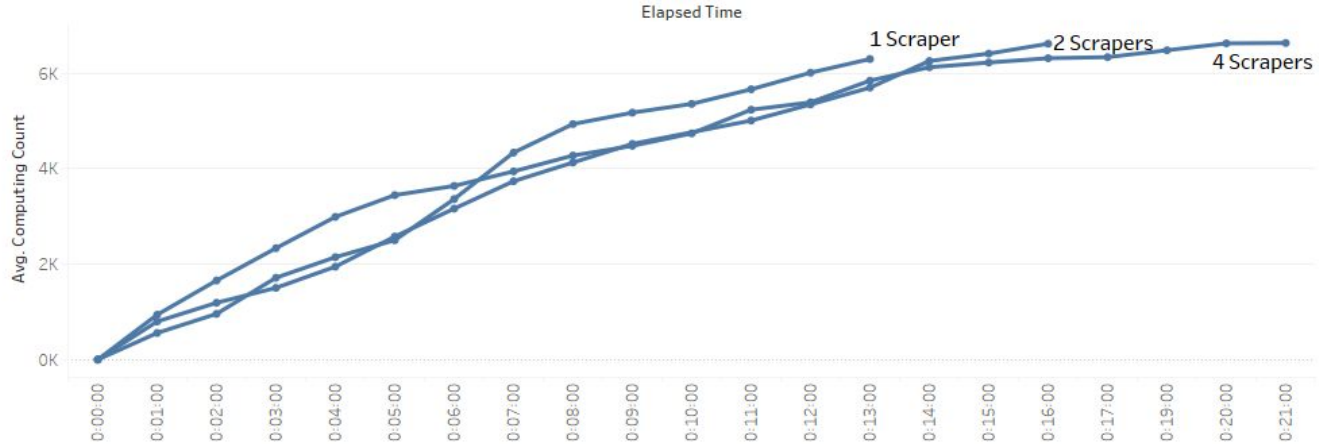
We see a positive correlation between higher depth settings/subprocesses and the number of webpages found containing a given keyword.

Note: This is only the results for the Computing keyword, but all showed the same trends.

Avg Keyword Found Over Time by Scrapers (10 initial, 25 filter, 45 max)



Avg Keyword Found Over Time by Scrapers (10 initial, 35 filter, 45 max)



Observations:

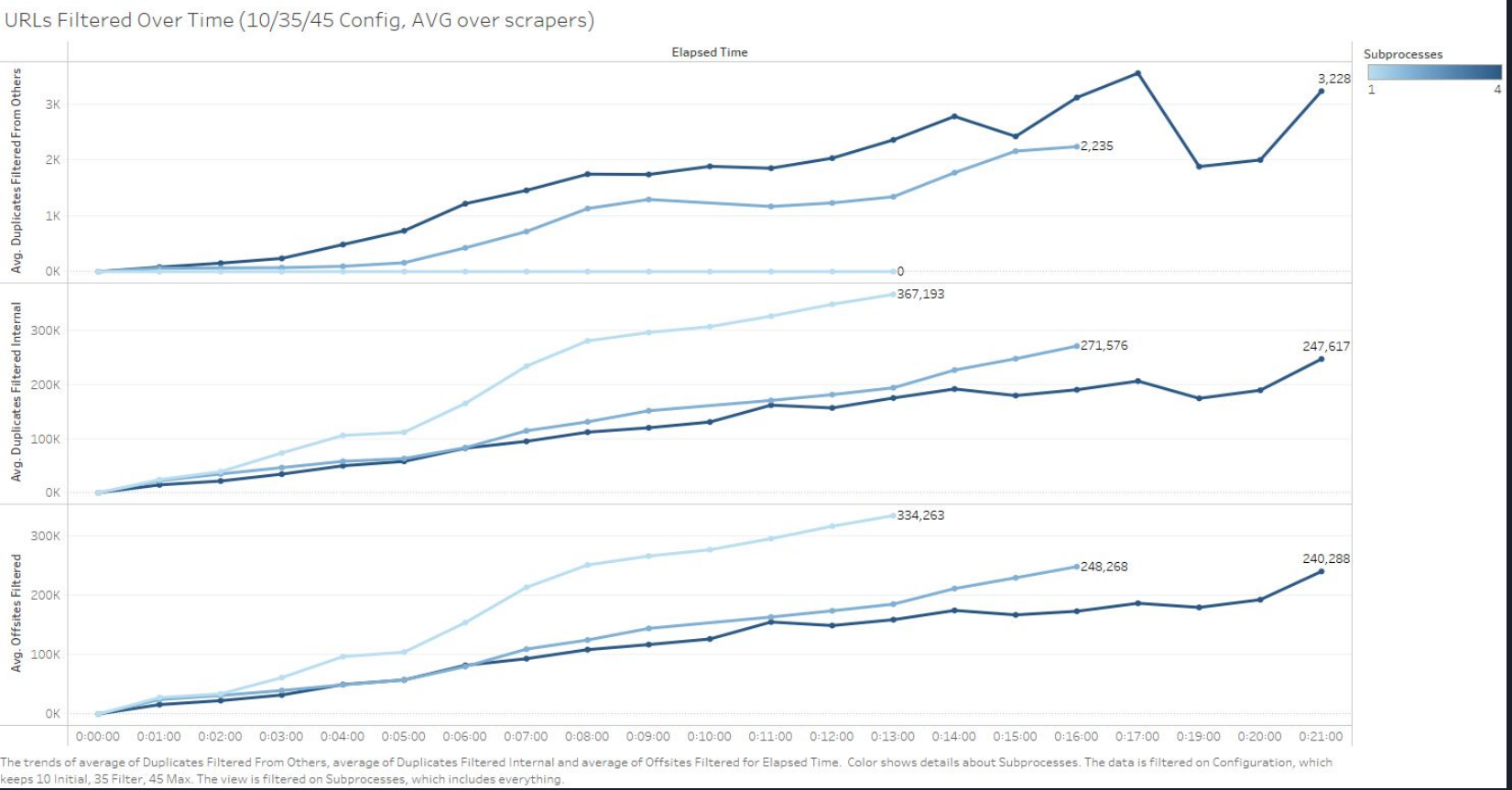
- The single scraper does worse with the lower filter depth
- The 10/35/45 configuration was consistently fastest for all configurations of scrapers

Crawling Results of Our Spider Configurations

To clarify the following graphs

Filtering behavior of spiders:

- **Filtered from others** = URLs filtered because another scraper has already processed it
- **Filtered internal** = URLs that the scraper itself has already encountered
- **Offsites filtered** = URLs that do not match the cc.gatech.edu/ criteria

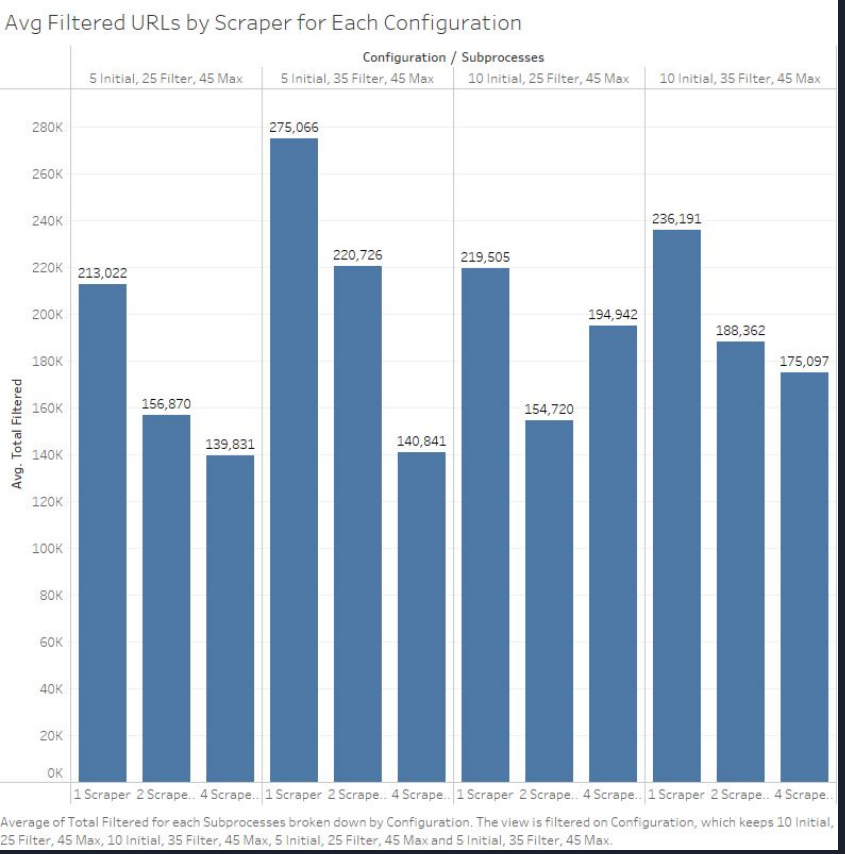


Trends:

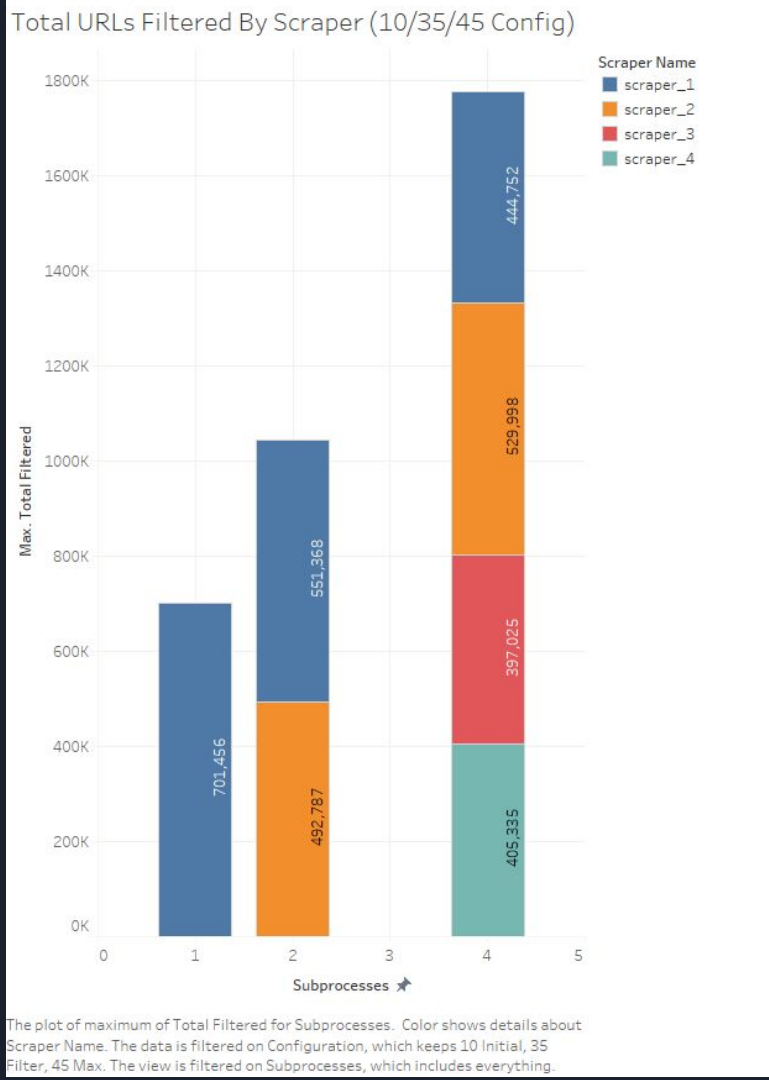
- One Scraper's number of filtered results quickly grows, showing a high number of URLs it filters
- Adding more scrapers results in filters less requests but takes longer

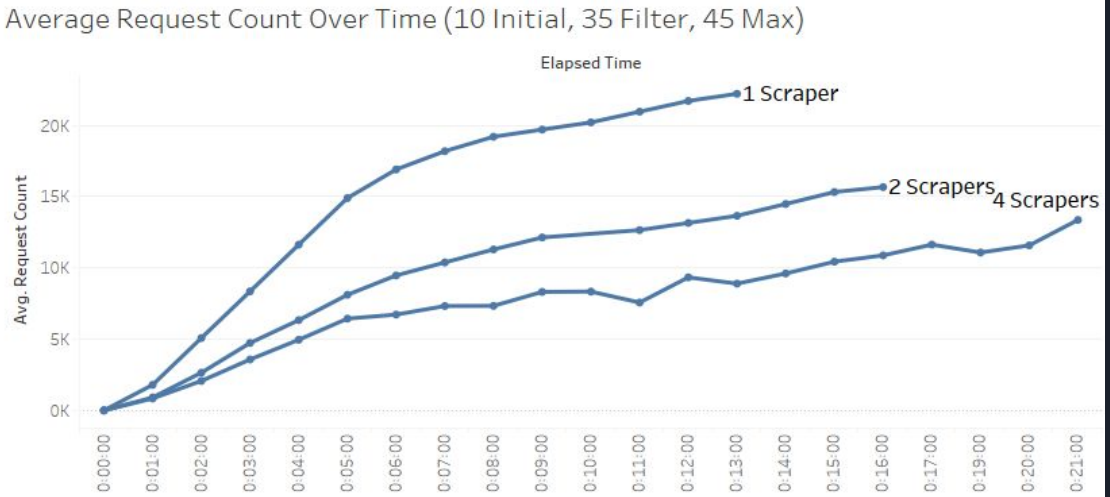
Possible Explanations:

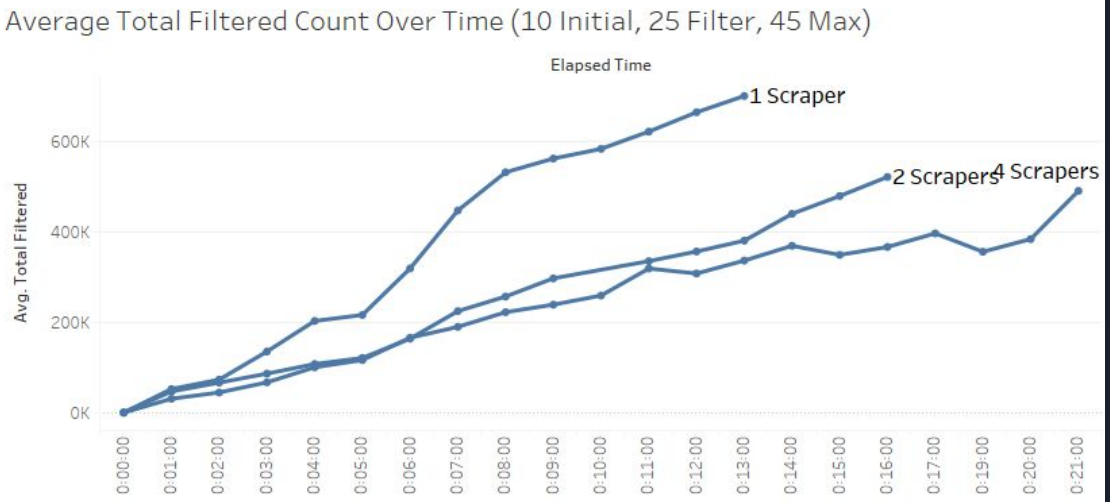
- Queries to check conflicts from other scrapers could be slowing down the scraper.



When we add more scrapers, each individual scraper filters less duplicates total, but adding all scrapers together shows that 4 scrapers is likely filtering the same URLs across multiple scrapers.

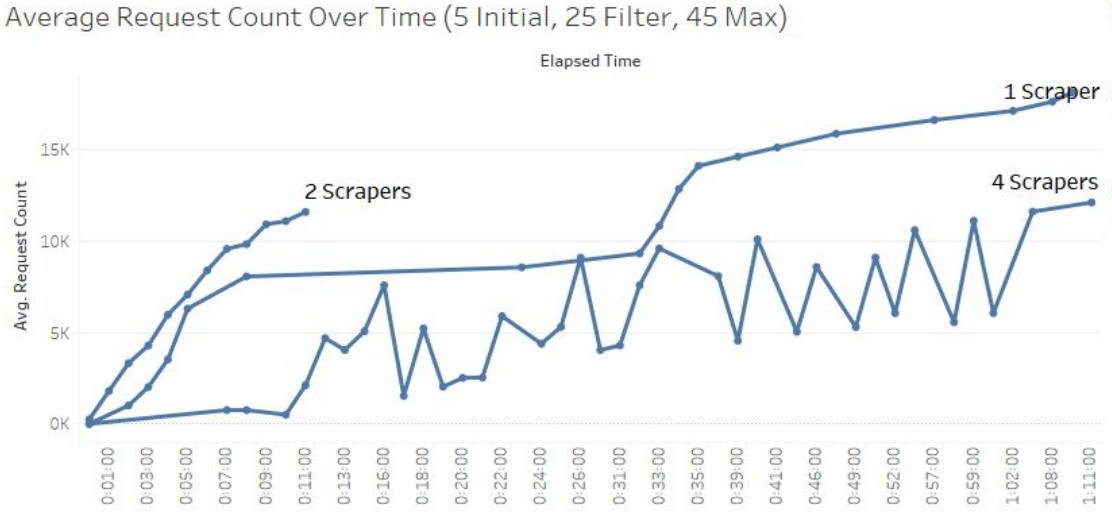


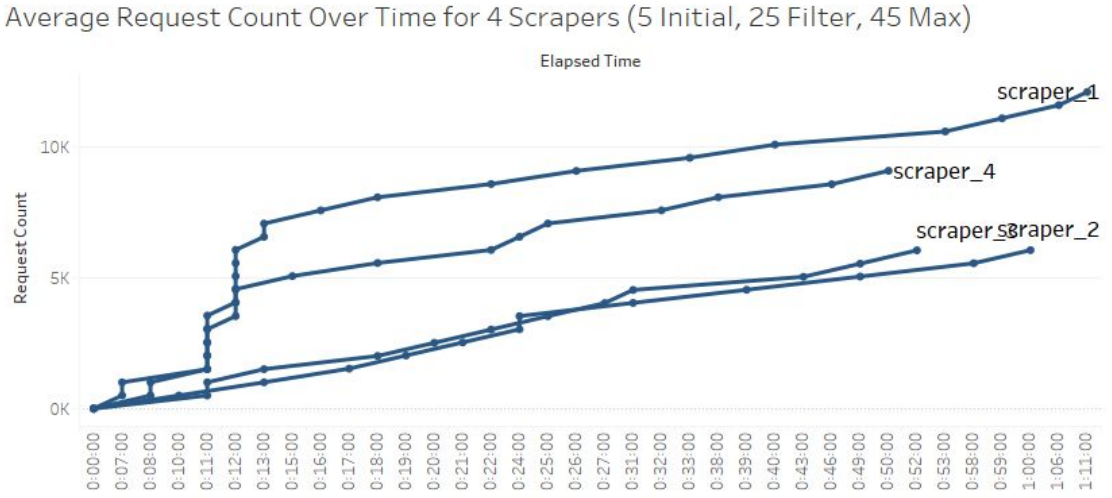




We see a mirroring of average requests made and average total filtered URLs indicicating:

- Filtering more URLs doesn't slow down requests.





Some of our data came out a bit sporadic, closer examination shows that some individual data points are missing, resulting in bad data on certain runs.

These runs and missing data points need to be investigated and conducted again.