Corona

Smruti R. Sarangi

Department of Computer Science
Indian Institute of Technology
New Delhi, India

Outline

1. Motivation
   - Motivation
   - Contemporary Approaches
   - Related Work
2. Corona Design
   - Design
   - Corona Schemes
   - Systems Management
3. Evaluation

Outline

1. Motivation
   - Motivation
   - Contemporary Approaches
   - Related Work
2. Corona Design
   - Design
   - Corona Schemes
   - Systems Management
3. Evaluation

Motivation

- The world wide web has a lot of information that changes frequently.
- There is no well defined publish-subscribe interface.
  - Publish : Post updates, and send to all the subscribers.
  - Subscribe : Notify the server, of the necessity to get updates.
- Polling based methods are not efficient.
- Corona provides a scalable method to disseminate updates.

Motivation - II

- Content that changes frequently
  - Blogs, wikis, news sites
- Current solution – micronews syndication
  - Based on naive polling .
- Polling – tradeoff between latency at the client and badwidth at the server.

Outline

1. Motivation
   - Motivation
   - Contemporary Approaches
   - Related Work
2. Corona Design
   - Design
   - Corona Schemes
   - Systems Management
3. Evaluation

Current Solutions

- Content providers impose rate limits based on – ip addresses, range of ip addresses.
- Servers ask the client to stop polling, or to change their polling intervals.
- Corona's aim:
  - Manage the server's bandwidth efficiently.
  - Stay within the limits.
  - Give the clients the best possible update latency.

Corona Front End

- Users subscribe by sending instant messages to a registered Corona userid.
- Corona ⇒ a cloud of nodes that monitors a set of channels
- A channel is a web page, or any other service that generates an active feed
- The Corona resource allocation algorithm dedicates a group of nodes to monitor each channel
  - They filter out useless content – timestamps, advertisements
  - A feed specific difference engine extracts the relevant portions that have changed
  - Distributes the changes to the clients

Outline

1. Motivation
   - Motivation
   - Contemporary Approaches
   - Related Work
2. Corona Design
   - Design
   - Corona Schemes
   - Systems Management
3. Evaluation

Background of Publish Subscribe Systems

- Publishers : Post content
- Subscribers : Subscribe to get relevant content
- Topic Based :
  - Publishers and subscribers are connected by a set of *topics*. Each topic is called a **channel**.
  - Subscribers get asynchronous updates over the channels.
- Content Based
  - Subscribers can make queries on the content, and receive results.
- Drawbacks of research prototype pub-sub systems: require custom interfaces, difficult to use
- Corona: backward compatible, easy to use (IM based)

Micronews Syndication

- Short updates of frequently changing data – news stories, blogs, facebook posts
  - Typically use an XML based format (examples – RSS, Atom)
- Accessed via http over standard URLs
- Use feed readers such as akregator to display data
- The server can inform the client when not to poll by using the *cloud* XML tag

RSS Example

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
<channel>
  <title>My  Home Page</title>
  <link>http://www.cse.iitd.ac.in/~srsarangi</link>
  <description>Homepage of S. R. Sarangi</description>
  <item>
    <title>Teaching</title>
    <link>teaching.html</link>
    <description>All teaching activities</description>
  </item>
  <item>
    <title>Research</title>
    <link>research.html </link>
    <description>Research Methodology </description>
  </item>
</channel>

</rss>
```

Outline

1. Motivation
   - Motivation
   - Contemporary Approaches
   - Related Work
2. Corona Design
   - Design
   - Corona Schemes
   - Systems Management
3. Evaluation

Corona – Cornell Online News Aggregator

- Key Features
  - Co-operative Polling – Assign multiple nodes to poll the same channel, and share updates.
  - Optimally distribute the task of polling
  - Corona poses this problem as an optimization problem, and solves it using the Honeycomb optimization toolkit

Design of Corona

- Corona uses a Pastry based overlay.
- Each channel has a unique channel identifier that is given a position along the Pastry ring.
- Corona defines a wedge around the channel that logically splits the set of nodes along the ring.
    - Wedge → A set of nodes sharing a common number of prefix digits with the channel's identifier.
- A channel has polling level *l*, if it is polled by all the nodes that have at least *l* matching prefix digits with it.
- A wedge associated with a channel polls for it.

Avg. Detection Time and Load on the Server

- A channel with polling level l, has on an average N/b^l nodes in its wedge
- Let *tau* be the polling interval.
  - Average detection time for updates $= \frac{\tau b^l}{2N}$
  - Collective load placed on the server $\propto \frac{N}{b^l}$

Problem

Problem: Estimate the polling levels of each channel, or, alternatively, the sizes of the wedges.

Outline

1. Motivation
   - Motivation
   - Contemporary Approaches
   - Related Work
2. Corona Design
   - Design
   - Corona Schemes
   - Systems Management
3. Evaluation

Corona-Lite

- Ensures good update performance, while ensuring that the load on the servers is light.

Optimization Problem

minimize $\left(\sum_1^M q_i \frac{b^{l_i}}{N}\right)$ such that $\left(\sum_1^M s_i \frac{N}{b^{l_i}} \leq \sum_1^M q_i\right)$

M	Number of channels
q_i	Number of clients for channel i
l_i	Polling level of channel i
N	Number of nodes
s_i	Content size for channel i

Corona-Lite - II

- Clients of popular channels gain a lot, because the average update time gets reduced.
- Nicely partitions bandwidth across channels.
- Update performance would vary depending on the type of the workload.

Corona-Fast

minimize $\left(\sum_1^M s_i \frac{N}{b^{l_i}}\right)$ such that $\left(\sum_1^M q_i \frac{b^{l_i}}{N} \leq T \sum_1^M q_i\right)$

M Number of channels

q_i Number of clients for channel i

l_i Polling level of channel i

N Number of nodes

s_i Content size for channel i

T Performance target

Aim

Minimize the load placed on the content servers, and achieve a target update time

Corona-Fast

- Bounds the total amount of network traffic.
- Allows us to tune the update performance per application. For example, a stock market application might choose a very fast update performance.
- Along with providing applications the desired level of update performance, it can shield web servers from spikes in network load.

Negative Aspects

- Both Corona-Lite and Corona-Fast do not consider the rate of change of objects in the channel.
- Corona-Fair takes this into account

Corona-Fair

$$\text{minimize } \left(\sum_1^M q_i \frac{\tau}{u_i} \frac{b^{l_i}}{N}\right) \text{ such that } \left(\sum_1^M s_i \frac{N}{b^{l_i}} \leq \sum_1^M q_i\right)$$

- M Number of channels
- q_i Number of clients for channel i
- l_i Polling level of channel i
- N Number of nodes
- s_i Content size for channel i
- T Performance target
- u_i Update interval for channel i
- τ Polling interval

Corona-Fair Sqrt and Log

Corona-Fair Sqrt

minimize $\left(\sum_1^M q_i \frac{\sqrt{\tau}}{\sqrt{u_i}} \frac{b^{l_i}}{N}\right)$ such that $\left(\sum_1^M s_i \frac{N}{b^{l_i}} \leq \sum_1^M q_i\right)$

Corona-Fair Log

minimize $\left(\sum_1^M q_i \frac{log(\tau)}{log(u_i)} \frac{b^{l_i}}{N}\right)$ such that $\left(\sum_1^M s_i \frac{N}{b^{l_i}} \leq \sum_1^M q_i\right)$

Corona-Fair

- Similar to Corona-Lite → minimizes update detection time, with a limit on the total amount of traffic
- Introduces a term to reduce the number of allocated servers if the rate of updates is small.
- It is possible to dampen this term by considering the square root or the log.

Decentralized Optimization

- Core optimization problem:

$$\text{min. } \sum_1^M f_i(l_i) \text{ s.t. } \sum_1^M g_i(l_i) \leq T$$

- f_i and g_i are the performance or the bandwidth cost of the channel at polling level l_i.
- The values of l_i are integers.
- This problem is NP-Hard (need to compute a fast approximation)
- Honeycomb finds a solution in $O(M \log M \log N)$ time, which is optimal for $M - 1$ channels.

Decentralized Optimization

- Honeycomb combines channels with similar tradeoffs into a tradeoff cluster.
- Honeycomb nodes periodically exchange these clusters.
- Periodically, nodes run the Honeycomb algorithm to figure out the assignment of nodes to channels.
- Disagreement regarding the assignment of nodes to channels can be a problem !!!

Outline

1. Motivation
   - Motivation
   - Contemporary Approaches
   - Related Work
2. Corona Design
   - Design
   - Corona Schemes
   - Systems Management
3. Evaluation

System Management

- Each channel in Corona hashes its content to get its unique Pastry key.
- It is assigned an owner node in Pastry.
- For added fault tolerance, a key is assigned to F succeeding nodes.
- Owners receive subscriptions, and send updates to the all the subscribers.
- Corona manages co-operative polling through three mechanisms:
  - Optimization Phase : Applies Honeycomb based optimization to the traffic data collected from servers.
  - Maintenance Phase : Changes to polling levels are communicated to peers.
  - Aggregation Phase : Nodes receive tradeoff data from other peers.

System Management - II

- Initially the owner nodes at $K = \lceil \log N \rceil$ poll for the channels.
- A node in the maintenance phase might decide to reduce the polling level to $K - 1$
- A small wedge will form that will perform co-operative polling
- When there is a change in the polling level, some nodes need to be instructed to start or stop polling.
- Owners typically monitor the status of the nodes in the wedge, and aggregate maintenance messages.
- Upon a failure Corona removes the node from the ring, and on the addition of a node, Corona adds it to the Pastry ring. If the owner fails, then Corona deletes its subscription state.

Update Dissemination

- Corona has a dedicated difference engine that computes the difference between different versions of a file by polling the server.
- It only sends the deltas (differences) to other nodes in the polling wedge.
- Each new version of a file has an unique version number.
- When a delta is generated by a node, it shares the delta with all the other nodes in the wedge.
- If a node cannot reliably get a timestamp from the server, then it sends the delta to the owner. The owner assigns a timestamp and multicasts it.

User Interface

- Users need to add Corona as a buddy in their IM system.
- They can then subscribe or unsubscribe to an URL by sending a message to Corona.
- A subscribe messages is routed to all the nodes in the polling wedge for that particular channel.
- When an update is detected by the owner of the channel, it is sent to all the subscribers through the IM system.
- IM systems typically allow peer to peer communication such as Skype.

Implementation

- Uses a standard Pastry implementation, 160-bit SHA-1 hash function
- Occasionally, it is possible that the size of a wedge might be zero
    - We need to then adjust the sizes of the clusters
- Corona interacts with IM systems using the instant messaging protocol - GAIM
- At the moment, the entire Corona system is trusted
- The evaluation is on a large scale deployment of Planet-Lab(large scale distributed cloud).
- Used a micronews feed collected from real life workloads.

Simulations

- System of 1024 nodes, 100,000 channels, and 5 million subscriptions
- Polling interval for 30 minutes, and maintenance interval of 1 hour
- Compare Corona-Lite, Corona-Fast, and Corona-Fair

Results

Scheme	Average Update Detection Time	Average Load
Legacy-RSS	900	50.00
Corona-Lite	53	48.97
Corona-Fair	142	50.14
Corona-Fair-Sqrt	55	49.46
Corona-Fair-Log	53	49.43
Corona-Fast	32	58.75

source [1]

Corona: A High Performance Publish-Subscribe System for the World Wide Web, Venugopal Ramasubramaniam, Ryan Peterson, and Emin Gun Sirer, NSDI 2006