**Introduction**

Rich Internet Applications (RIAs) allow better user interaction and responsiveness than traditional web applications. Thanks to new technologies like AJAX (Asynchronous JavaScript and XML), Rich Internet Applications can communicate with the server asynchronously. This allows continuous user interactions.

Security of RIA and automating security testing are important, ongoing, and growing concerns. One important aspect of this automation is the crawling of RIAs, i.e., reaching all possible states of the application from the initial state. Being able to do so automatically is also valuable for search engines and accessibility assessment.

Crawling of RIA applications is an expensive and time-consuming process due to their large number of states. To accelerate this operation, we distribute the operation over many nodes in an elastic cloud environment.

**Background**

- **Crawling Strategy:**
  - Breath-First Search
  - Bounded Depth-First Search
  - Based on page weight
  - Model Based Crawling
  - Hypercube Model
  - Menu Model
  - Greedy algorithm
  - Probabilistic model

- **Partitioning strategies:** Mostly use server-related matrix as primary tool to partition the search space:
  - Page URL
  - Server IP address
  - Server geographical location
  - [Loo 2004] describes distributed web crawling by hashing the URL

**Proposed Architecture**

- Nodes act autonomously and independently. Each node starts at an initial state, then gets tasks to go to Active state, when there is nothing to do and goes to idle state, and finally terminates when the termination order arrives.

- A virtual ring is created based on breath-first-search traversal of the nodes. A termination token goes around this ring that keeps the list of states IDs and the number of states visited by each node. When all states are visited on all nodes, a termination order is broadcasted.

**Motivation and Aim**

- **Non-URL-Based Crawling strategy:**
  - In a RIA, one URL corresponds to many states of DOM. Unlike traditional websites in which every call to server would change the whole DOM and the page URL, RIA relies on small AJAX updates that do not necessarily modify the page URL:
    - Traditional distributed crawlers rely heavily on URLs, in order to partition the search space. Underlying assumption for this strategy is a one-to-one correspondence between the URL and the state of DOM which does not hold in RIA.
    - Therefore, we propose to partition the search space based on events.
  - **Crawling Strategy:** Reduce the workload by choosing the events to execute using Greedy algorithm.
  - **Crawling Efficiency:** Discover states as soon as possible, using Probabilistic model.

References:
- [Dincturk 2014] Dincturk, Mustaffa Eme and Jourdan, Guy-Vincent and Bochmann, Gregor von and Onut, Iosif Viorel: Model Based Crawling, ACM Transactions on the WEB

**Algorithm**

- Algorithm 1 Crawling Algorithm (As Executed at Each Node)

```plaintext
if (NODESTATUS == ACTIVE) then
  GET(INITIALNODE, FROMCOORDINATOR)
else
  if (NOT NODESTATUS is not TERMINATED) then
    WORKINGSTATE ← TERMINATE
    if (WORKINGSTATE is Empty) then
      NODESTATUS ← DONE
    else
      NODESTATUS ← ACTIVE
    end if
  end if
  if (STATE TO VISIT is Empty) then
    PICK(STATE, (WORKINGSTATE)
    EXECUTE(STATE, TO VISIT, EVENT TO EXECUTE)
    if (CURRENTSTATE is not in DISCOVEREDSTATES) then
      push CURRENTSTATE to DISCOVEREDSTATES
    end if
    if (STATE TO VISIT is Empty) then
      REMOVED(STATE, TERMINATE)
    end if
  end if
end if
```

**Experimental Results**

- **Crawling Efficiency** measures how early in the crawl new states are discovered.
  - Distributed Greedy algorithm has the best performance in terms of total time it takes to crawl a website.
  - Distributed Probabilistic model is the most efficient algorithm and discovers states early in the crawl.

**Future Work**

- We are currently working on distributed crawling of RIA in a cloud environment.
- We plan to add fault tolerance to our strategy so that if some of the nodes crash, the rest of the nodes continue without interruption.
- Once we have a working implementation of the system, we plan to optimize it based on different infrastructure parameters such as cost of communication or the processing power available to different nodes.

**Acknowledgments**

This work is supported in part by Center for Advanced Studies, IBM Canada.