Parallel Crawlers

AND

Efficient URL Caching for World Wide Web Crawling

Presenter
Sawood Alam
salam@cs.odu.edu
Parallel Crawlers

Junghoo Cho
University of California
cho@cs.ucla.edu

Hector Garcia-Molina
Stanford University
cho@cs.stanford.edu
ABSTRACT

- Design an effective and scalable parallel crawler
- Propose multiple architectures for a parallel crawler
- Identify fundamental issues related to parallel crawling
- Metrics to evaluate a parallel crawler
- Compare the proposed architectures using 40 million pages
Challenges for parallel crawlers

- Overlap
- Quality
- Communication bandwidth
Advantages

- Scalability
- Network-load dispersion
- Network-load reduction
  - Compression
  - Difference
  - Summarization
Related work

- General architecture
- Page selection
- Page update
Geographical categorization

- Intra-site parallel crawler
- Distributed crawler

Figure 1: General architecture of a parallel crawler
Communication

- Independent
- Dynamic assignment
- Static assignment

Figure 2: Site $S_1$ is crawled by $C_1$ and site $S_2$ is crawled by $C_2$
Crawling modes (Static)

- Firewall mode
- Cross-over mode
- Exchange mode

Figure 3: Summary of the options discussed
URL exchange minimization

- Batch communication
- Replication
Partitioning function

- URL-hash based
- Site-hash based
- Hierarchical
Evaluation models

- Overlap
- Coverage
- Quality
- Communication overhead

<table>
<thead>
<tr>
<th>Mode</th>
<th>Coverage</th>
<th>Overlap</th>
<th>Quality</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall</td>
<td>Bad</td>
<td>Good</td>
<td>Bad</td>
<td>Good</td>
</tr>
<tr>
<td>Cross-over</td>
<td>Good</td>
<td>Bad</td>
<td>Bad</td>
<td>Good</td>
</tr>
<tr>
<td>Exchange</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Bad</td>
</tr>
</tbody>
</table>
Firewall mode and coverage

Figure 4: Number of processes vs. Coverage

Figure 5: Number of seed URLs vs. Coverage
Cross-over mode and overlap

Figure 6: Coverage vs. Overlap for a cross-over mode crawler
Exchange mode and communication

Figure 7: Number of crawling processes vs. Number of URLs exchanged per page
Quality and batch communication

Figure 8: Crawlers downloaded 500K pages (1.2% of 40M)  

Figure 9: Crawlers downloaded 2M pages (5% of 40M)  

Figure 10: Crawlers downloaded 8M pages (20% of 40M)
Conclusion

- Firewall mode is good if processes $\leq 4$
- URL exchange poses network overhead $< 1\%$
- Quality is maintained even in the batch communication
- Replicating 10,000 to 100,000 popular URLs can reduce 40% communication overhead
Introduction

- Fetch a page
- Parse it to extract all linked URLs
- For all the URLs not seen before, repeat the process
Challenges

- The web is very large (coverage)
  - doubling every 9-12 months
- Web pages are changing rapidly (freshness)
  - all changes (40% weekly)
  - changes by a third or more (7% weekly)
Crawlers

- IA crawler
- Original Google crawler
- Mercator web crawler
- Cho and Garcia-Molina’s crawler
- WebFountain
- UbiCrawler
- Shkapenyuk and Suel’s crawler
Caching

- Analogous to OS cache
- Non-uniformity of requests
- Temporal correlation or locality of reference
Caching algorithms

- Infinite cache (INFINITE)
- Clairvoyant caching (MIN)
- Least recently used (LRU)
- CLOCK
- Random replacement (RANDOM)
- Static caching (STATIC)
Experimental setup

Figure 1: The flow of URLs and pages through our Mercator setup
URL Streams

• full trace
• cross sub-trace
Result plots

Figure 2: Miss rate as a function of cache size for the full trace

Figure 3: Relative miss rate (MIN = 1) as a function of cache size for the full trace

Figure 4: Miss rate as a function of cache size for the cross sub-trace

Figure 5: Relative miss rate (MIN = 1) as a function of cache size for the cross sub-trace
Result plots

Figure 6: Miss rate as a function of cache size for the full trace for various numbers of threads (using CLOCK)

Figure 7: Miss rate as a function of cache size for the cross sub-trace for various numbers of threads (using CLOCK)

Figure 8: Miss rate as a function of cache size per thread for the full trace

Figure 9: Miss rate as a function of cache size per thread for the cross sub-trace
Result plots

Figure 10: Miss rate as a function of time for the full trace (Cache size = $2^{18}$)

Figure 11: Miss rate as a function of time for the cross sub-trace (Cache size = $2^{18}$)
Results

- LRU & CLOCK performed equally well but slightly worse than MIN except for critical region (for both traces)
- RANDOM is slightly inferior to CLOCK and LRU, while STATIC is generally much worse
- Concludes considerable locality of reference in the traces
- For very large cache STATIC is better than MIN (excluding initial k misses)
- STATIC is relatively better for cross trace
  - Lack of deep links, often pointing to home pages.
  - Intersection between the most popular URLs and the cross trace tends to be larger
Critical region

- Miss rate for all efficient algorithms is constant (~70%) in $k = 2^{14} - 2^{18}$
- Above $k = 2^{18}$ miss rate drops abruptly to ~20%
Cache Implementation

Figure 12: Hashing with direct chaining

Figure 13: Hashing with direct circular chaining

Figure 14: Scatter table with circular lists
Conclusions and future directions

- 1,800 simulations over 26.86 billion URLs resulted cache of 50,000 entries gives 80% hit rate
- Cache size of 100 ~ 500 entries per thread is recommended
- CLOCK or RANDOM implementation using scatter table with circular chain is recommended
- To what order graph traversal method affects caching?
- Global cache or per thread cache is better?
THANKS