Publishing System for Efficiently creating dynamic Web Content

- Jim Challenger, Arun Iyengar, Karen Witting, Cameron Ferstat, Paul Reed

Anagha Gupte CPS 296
March 5, 2002

Problems that arise with Dynamic Data

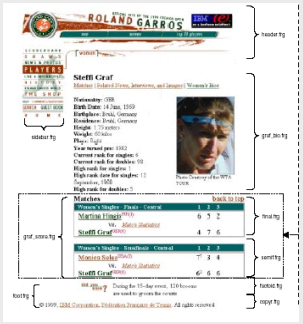
- Higher overhead than in static data.
- Complete and consistent updates of all pages important at all times.
- Identification of which cached pages have been affected by changes to underlying data is difficult.

This paper comes up with a solution for efficiently and consistently publishing dynamic Web content.

What does the system do?

- Implemented by IBM Research.
- It efficiently creates dynamic web content.
- It manages complex Web pages composed of fragments
- Complex objects constructed from fragments which may recursively embed other fragments.
- Algorithms for automatically and efficiently detecting and updating Web pages affected after fragments change are presented here.
- It has been implemented in both Java and C++.

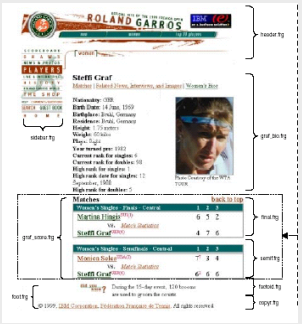
Our example



This system was used for the official site of the 1999 French Open Tennis Tournament.

We will try to explain the system using this page as an example.

Our example



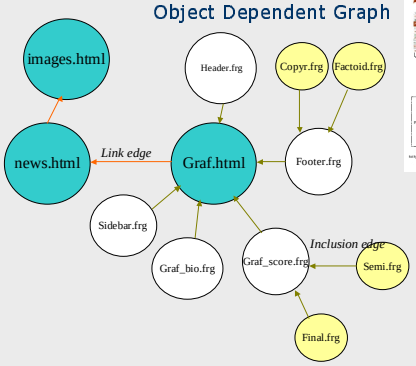
We will talk about the following:

- System Architecture
- System Description
- System Performance

Terminology

- A page or a fragment known as an object.
- *Atomic page/fragment:* one that doesn't include any other fragments
- *Complex page/ fragment:* includes other fragments
- Inclusion relationships among fragments and web pages are represented by an *ODG (Object Dependent Graph)*
- *Combined content page:* includes both human and computer- generated information
- *Immediate fragments:* contain vital information- need to be updated ASAP
- *Quality controlled fragments:* contain content that needs to be examined before publishing- don't have to be published in a hurry
- *Bundle:* pages published together in an atomic action

System Architecture
Object Dependent Graph



Incremental Publication

- Required because of immediate and quality controlled fragments.
- Information is published in stages.
- Doesn't guarantee complete consistency.
- 3 methods of incremental publication provided.

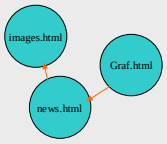
Incremental Publication

Method 1

The first method guarantees that a newly published page will not contain a link to an unpublished page or to a non- existent link.

Ex. R ={graf.html, news.html, images.html}

(sub graph)



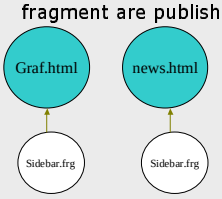
Make sure that all hypertext links on a page are updated before that page is updated.

The links should not be obsolete as compared to our page neither should they be broken.

Incremental Publication

Method 2

The second method guarantees that any two pages that contain a common changed fragment are published in the same fragment.

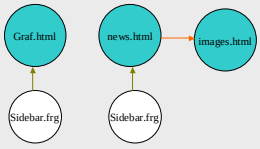


Since sidebar.frg is present in both graf.html and in news.html, both these pages must be published in the same bundle.

Incremental Publication

Method 3

- The third method satisfies consistency guarantees of the first 2 methods.

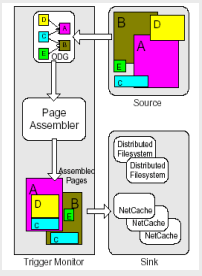


News.html contains a hypertext link to images.html so the updated version of news.html cannot be published before images.html is updated.

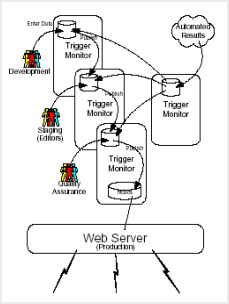
If images.html and sidebar.frg change, then updated versions of graf.html and news.html must be published together since they contain sidebar.frg.
So all pages published in one bundle.

System Description

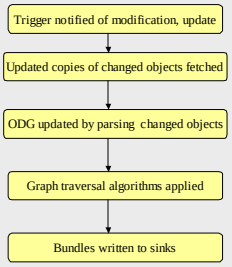
Schematic of the Publish Process



Stages of Trigger Monitor Software



- *Development:* involves overall site design- a structurally complete site
- *Staging:* Result is a finished page
- *Quality Assurance:* pages examined for correctness and appropriateness.
- *Automated Results:* produced on update trigger. No human intervention here.

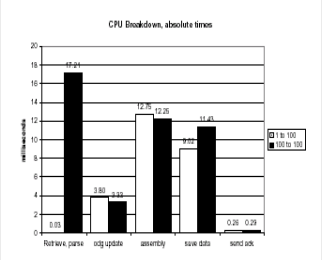


- *Production:* pages served from here. It is a source for the serving directories and caches.

System Performance

- Web page size - 10Kbytes; 100 bytes < Fragment size < 1Kbyte
- CPU time consumed by the following activities:
  - *Retrieving, parsing:* reading triggered objects from disk; parsing to determine included fragments.
  - *ODG updating:* analyzing & updating the ODG based on parsing information
  - *Assembling:* updating all objects
  - *Saving data:* saving updated objects to disk
  - *Sending acknowledgements:* publication completion acknowledgement via HTTP.

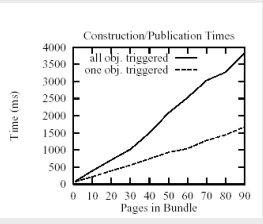
Breakdown in CPU time required to construct & publish a typical complex Web page



ODG information can be fetched from memory as opposed to retrieval and parsing the entire fragment information, so OD graph traversal has low overhead

Due to concurrency in object construction, near 100% CPU utilization is achieved.

CPU Time required to construct bundles of various sizes



The curves are proportional

- However, CPU time lesser when one object triggered

Conclusion from graph:

- Better to trigger fewer objects that are present in multiple pages than to trigger all objects which need to be constructed.

Conclusion

- Objects created from fragments that recursively embed other fragments
- All pages associated with updated fragments have to be re- published each time
- Different incremental publishing algorithms to handle different consistency requirements
- Trigger can take input from various sources
- Multiple authors can update the content at one time

To think about…

- If Immediate and Quality Controlled fragments are updated together, there is considerable delay in publishing
- The solution for this is incremental publishing…but it doesn't guarantee consistency
- Should fragments be cached and pulled out on request? Or is the push mechanism used here better? It would have higher overhead than pull.
- Trade off between freshness and update overhead?
- Is Trigger Monitoring software the optimal solution or are there better solutions to be found?