Fast Browsing of Archived Web Contents

Sangchul Song and Jeseph JaJa

Institute for Advanced Computer Science Studies
Department of Electrical and Computer Engineering
University of Maryland, College Park, Maryland, USA

Contents

Motivation and Goal

Our Approach

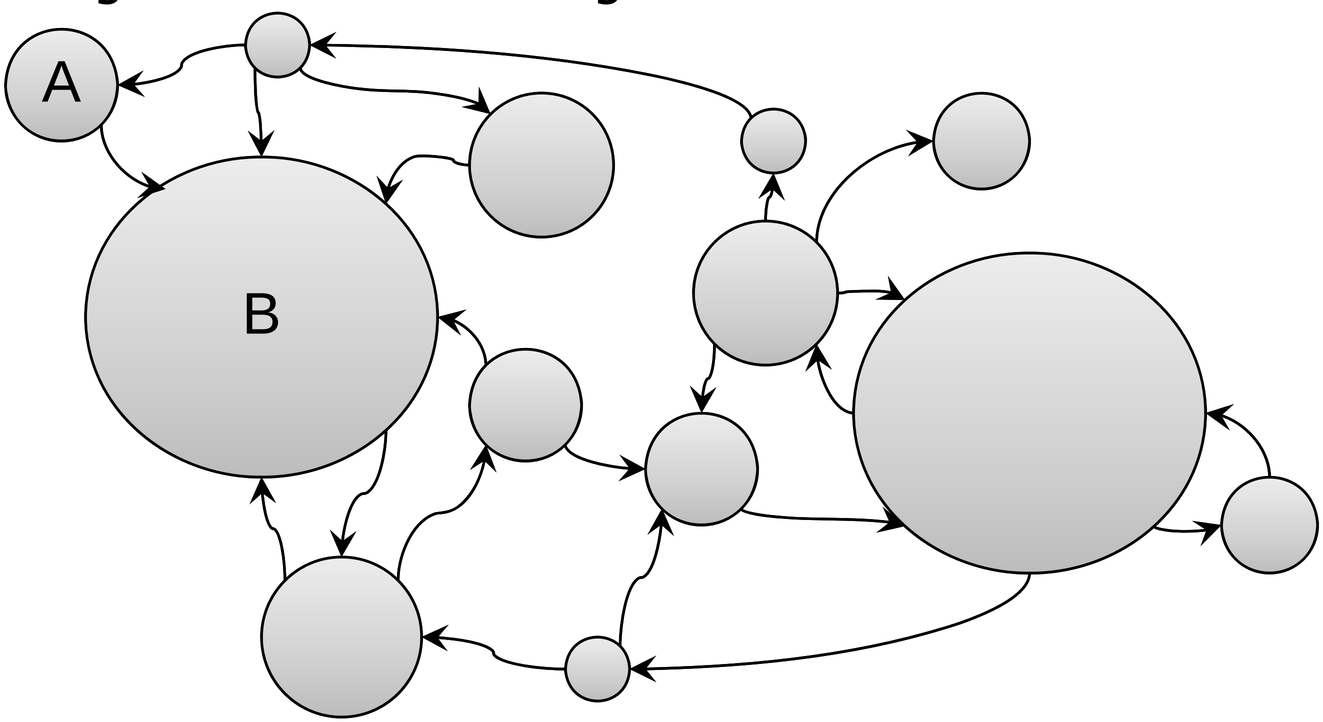
Experiments

Conclusions

Motivation and Goal

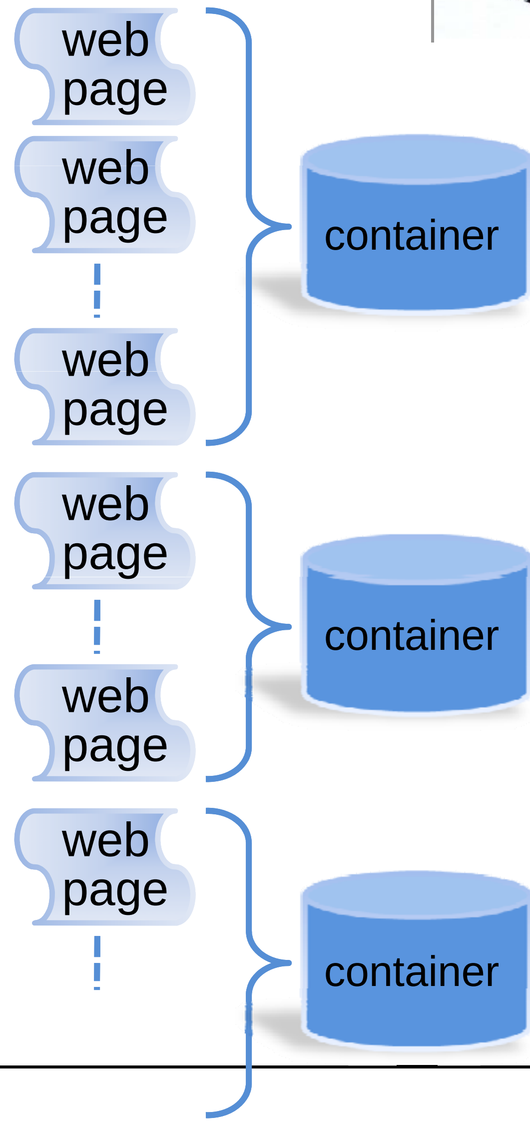
Web Graph

- Web as a weighted, directed graph.
  - Web page → Vertex
  - Hyperlink → Edge
  - Page size → Vertex weight



Web Containers

- A multitude of web pages are aggregated into a container (e.g. WARC)
- Each container serves as an archive object
- Currently adopted by the Internet Archive, and many other national archives, and libraries



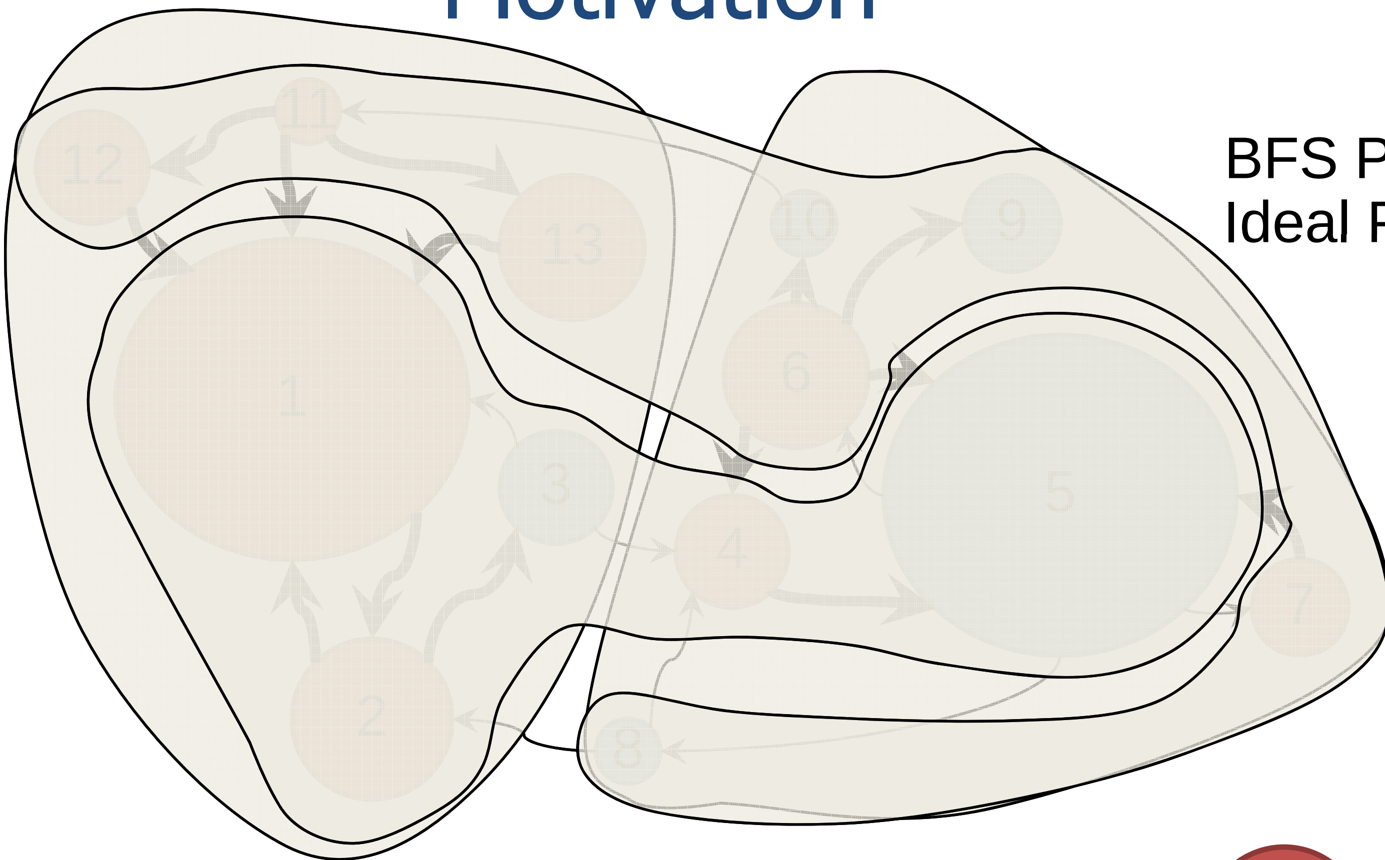
Current Container Packaging

- BFS or DFS based crawls
- First seen page → first put into a container
- When a container is full, a new container is created.

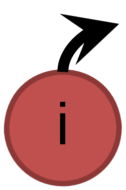
```
Input
Seed URLs : {url_1, url_2, … }
MAX_SIZE

Procedure
Enqueue(Q, Seed URLs)
i←1
visited[] ← FALSE
C_i ← new Container()
while (Q is non-empty)
        u ← Dequeue(Q)
        Fetch(u);
        visited[u] ← TRUE
        if (Size(C_i) + Size(u) > MAX_SIZE)
                i = i + 1
                C_i = new Container()
        C_i = C_i  U u
        for each v ∈ adjacent[u]
                if (visited[u] = FALSE)
                        Enqueue(Q, v)
```

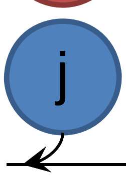
Motivation



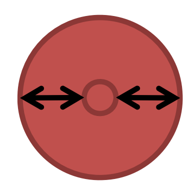
BFS Packaging
Ideal Packaging



Popular Page (visited i_{th} by BFS)



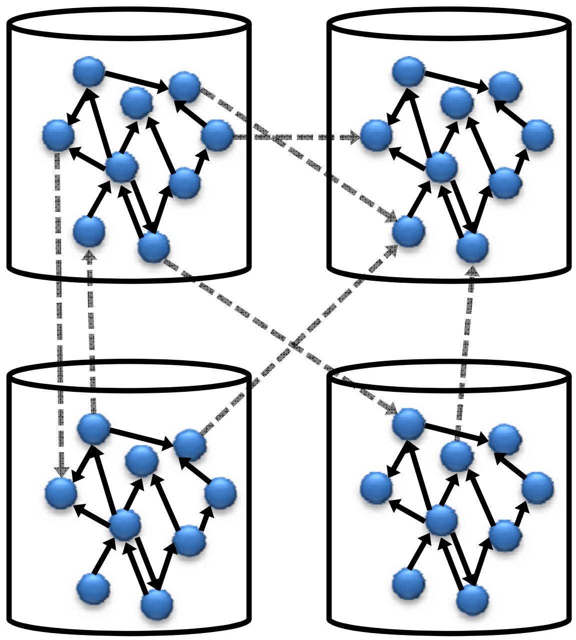
Unpopular Page (visited j_{th} by BFS)



Circle Size ≈ Page Size

Our Goal

- Minimize the probability that a user jumps often between containers.



Packaging Issues

Main Issue: How to minimize the number of containers necessary when accessing the archived web pages?

How to put together more relevant web pages in the same container?
→ Graph Partitioning

How to define 'relevancy'?
→ Graph Analysis

Our Approach

Our Approach

- Graph Analysis
  - Web graph is analyzed to obtain, for each edge, a global probability that the edge is taken.
- Graph Partitioning
  - Using edge weights from graph analysis, find the best partition where the sum of edge weights across different parts is minimized, and the size of each part is balanced.

```
Input
Seed URLs : {url_1, url_2, … }
MAX_SIZE

Procedure
G ← BuildWebGraph(Seed URLs)
n ← GetNumberOfContainers(G,MAX_SIZE)
G ← EdgeRank(G) /* Optional */
{UL_1,UL_2,…,UL_n} ← PartitionGraph(G, n)
for ( 1 ≤ i ≤ n)
    C_i ← new Container()
    for (v ∈ UL_n)
        fetch(v)
        C_i = C_i U v
```

Graph Analysis (1/2)

- PageRank [Brin, S. and Page, L, 1998]
  - Computes the importance of each web page.
  - Pages linked from important pages are considered important.
  - Ideal model: $PR(u) = \sum_{v \in I_u} p_{vu} PR(v)$
  - Two problems.
    - Dangling pages ➔ Solution: artificial out-links to all the other pages from dangling pages, w/ probability of *1/N*
    - Cyclic paths ➔ Solution: artificial out-links to all the other pages from each page, w/ probability of *1-d*
  - Modified model: $PR(u) = \frac{1-d}{N} + d \sum_{v \in I_u} p_{vu} PR(v)$

Graph Analysis (2/2)

- EdgeRank

$$ER(e) = \frac{PR(v)}{\text{outdegree}(v)}$$

- EdgeRank is used in our simulation

Graph Partitioning (1/2)

- Edge-Cut : The sum of the weights of the edges that connect any two different parts.
- *Web Graph Partitioning Problem*: Given a directed web graph $G:(V, E)$ with weighted nodes (weight of a node is the size of the corresponding page) and weighted edges, determine a partition $V = P_1 \cup P_2 \cup P_3 \cup \ldots \cup P_n$ such that,
  1. Edge-Cut is minimized.
  2. For all i's, $|P_i| \leq K$ for some fixed K, where $|P_i|$ is the sum of the weights of the vertices in P_i and K is an upper bound on the size of a container.
- This is an NP-Complete problem

Graph Partitioning (2/2)

- Scheme used in simulation: [Karypis and Kumar, 1998]
  - Fast multilevel graph partitioning algorithm
    1. First compute a maximal matching using a randomized algorithm
    2. Coarsen the graph by collapsing the matched vertices together
    3. Repeat 1~2 until a desired size of the coarsened graph is achieved
    4. Compute minimum edge-cut bisection
    5. Refine & uncoarsen the partitioned graph.

Experiments

Experiment Settings (1/2)

- Datasets
  - UMIACS : Web Graph built from our crawls of http://umiacs.umd.edu in 2007
  - Stanford : Web Graph built from a crawl of http://stanford.edu in 2002 by the Stanford WebBase project

- Dataset Properties

Datasets	# Vertices	# Edges	Total Vertex Weight
UMIACS Web Graph	4579	9732	2.49GB
Stanford Web Graph	281903	2312497	215.82GB

Experiment Settings (2/2)

- Three packaging methods
  - CONV: Pages are allocated to containers as they are fetched during the crawling process (BFS). Once a container is full, we use a new container.
  - GP: The graph partitioning technique is applied so as to minimize the number of edges connecting any two partitions. All the pages belonging to a partition are allocated to a single container.
  - ER+GP: EdgeRank is used to assign weights to edges, and the graph is partitioned using a minimum-weight partitioning algorithm.

Edge-Cut Result

$$EC_{scaled} = \frac{EC \times 100}{|E|},$$

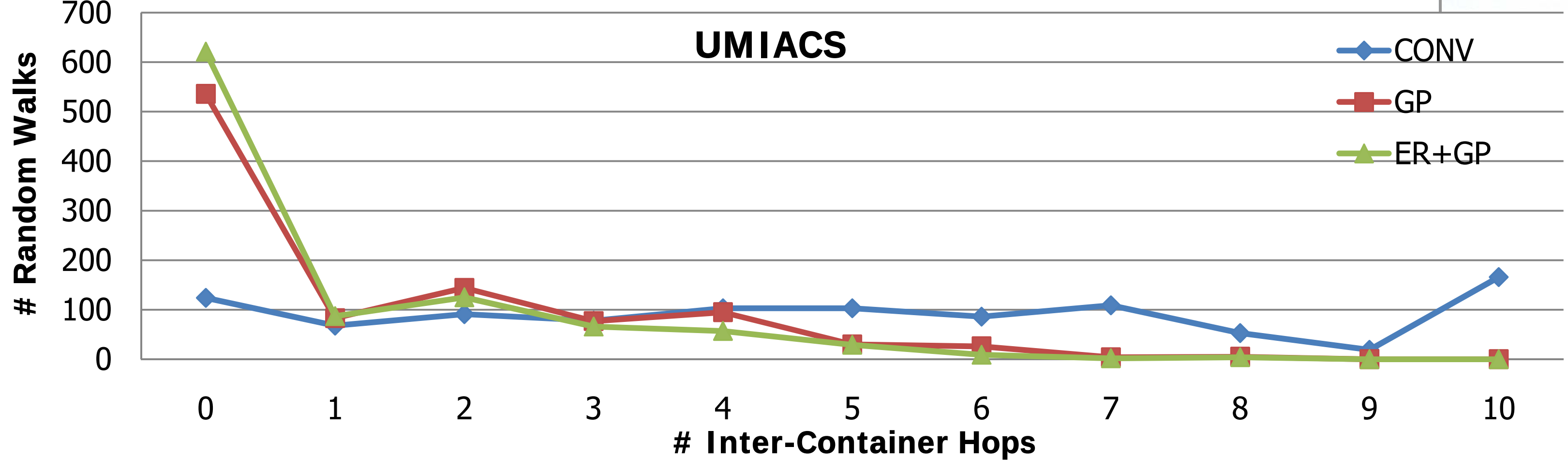
Edge-Cut	Unweighted Edges		Weighted Edges	
	CONV	**GP**	**ER+CONV**	**ER+GP**
UMIACS Web Graph	73.87	12.38	62.36	36.03
Stanford Web Graph	80.50	47.33	63.56	32.20

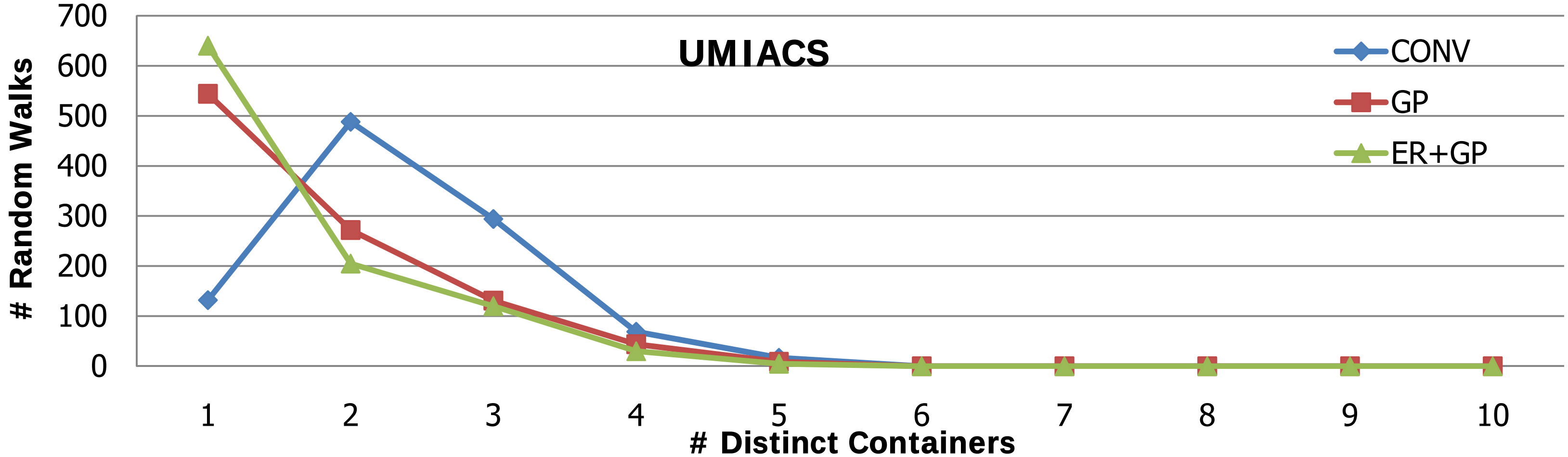
Simulation

- Simulation Parameters

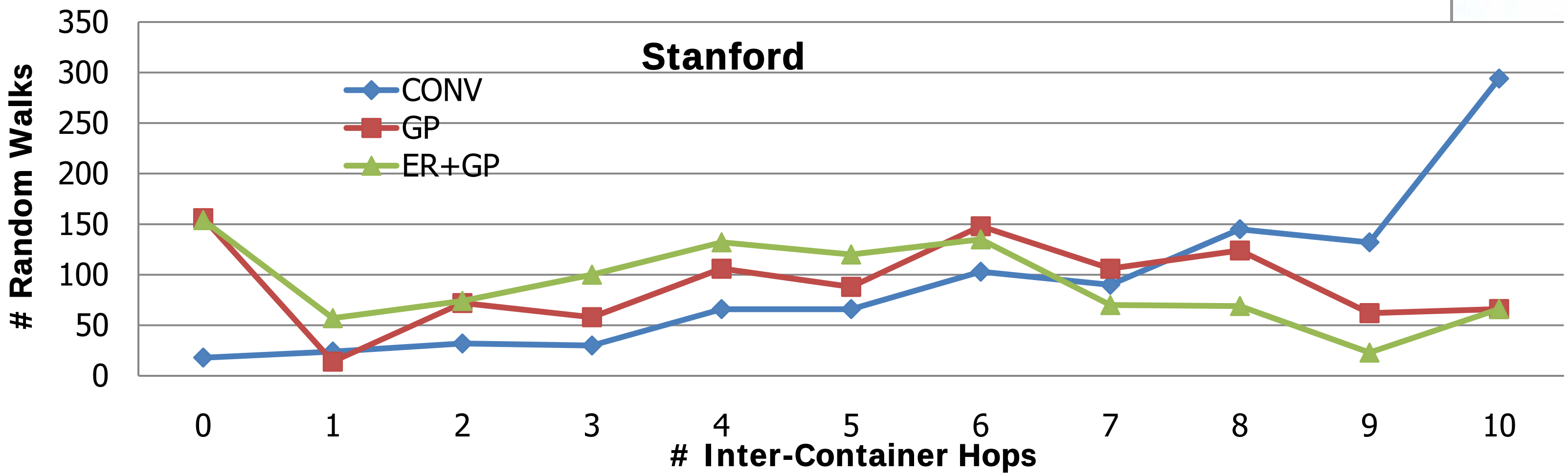
Parameter	Value
Number of Random Walks	1000
Number of Hops in Each Walk	10
Probability of Going Back	30%
Out-degree of Starting Vertex	> 5
Policy At Dangling Vertex	Go back

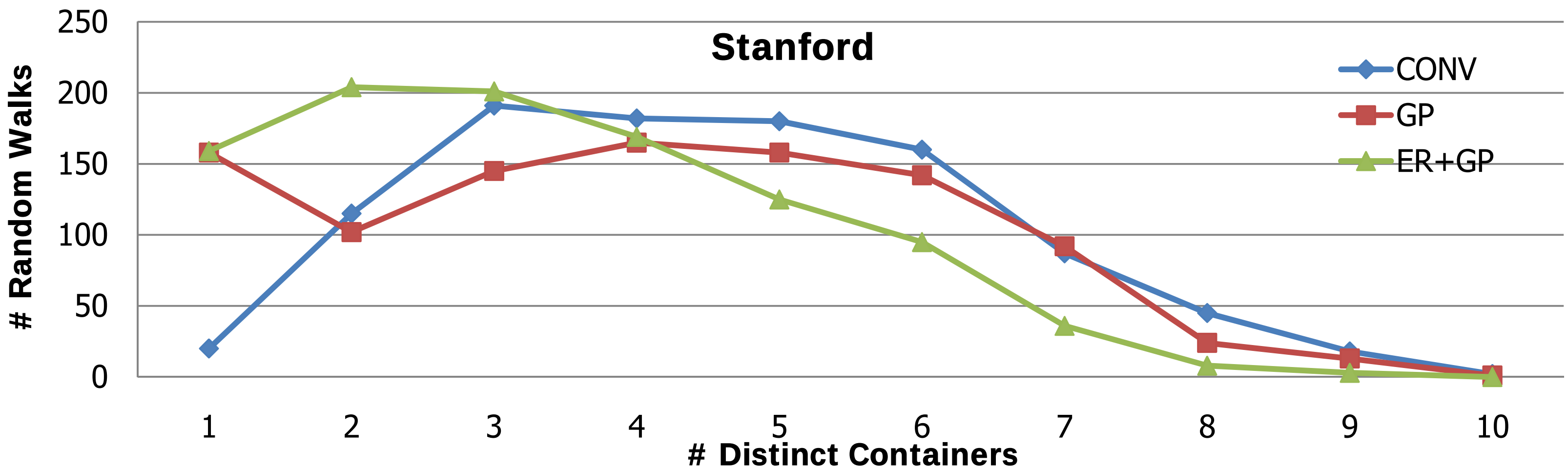
Simulation Results (UMIACS)



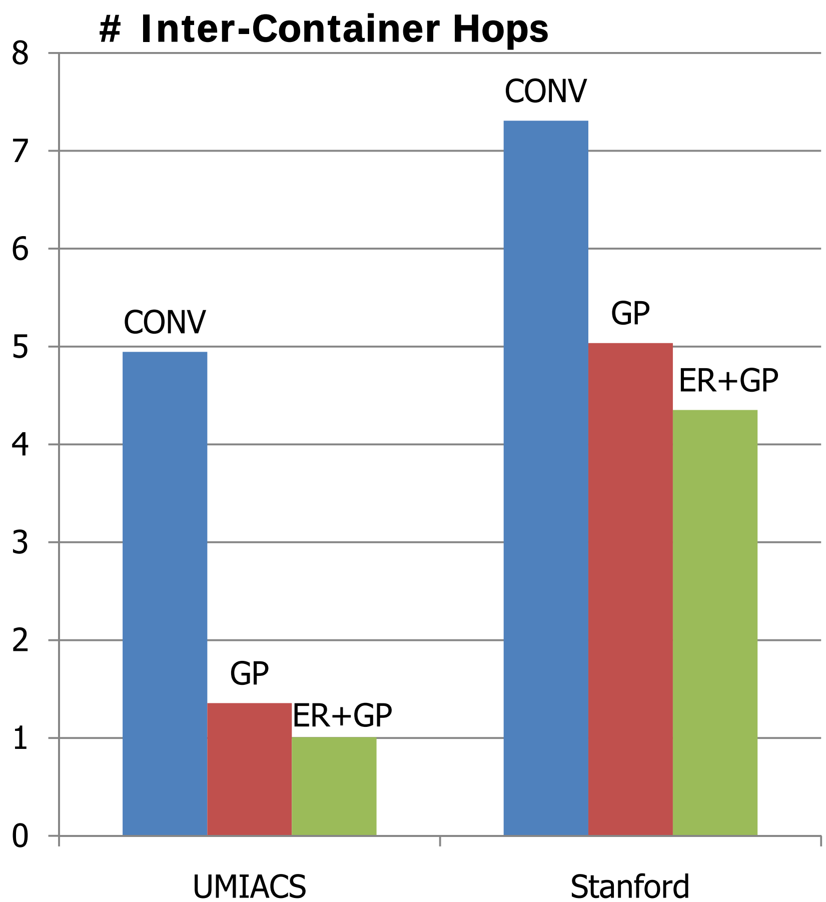


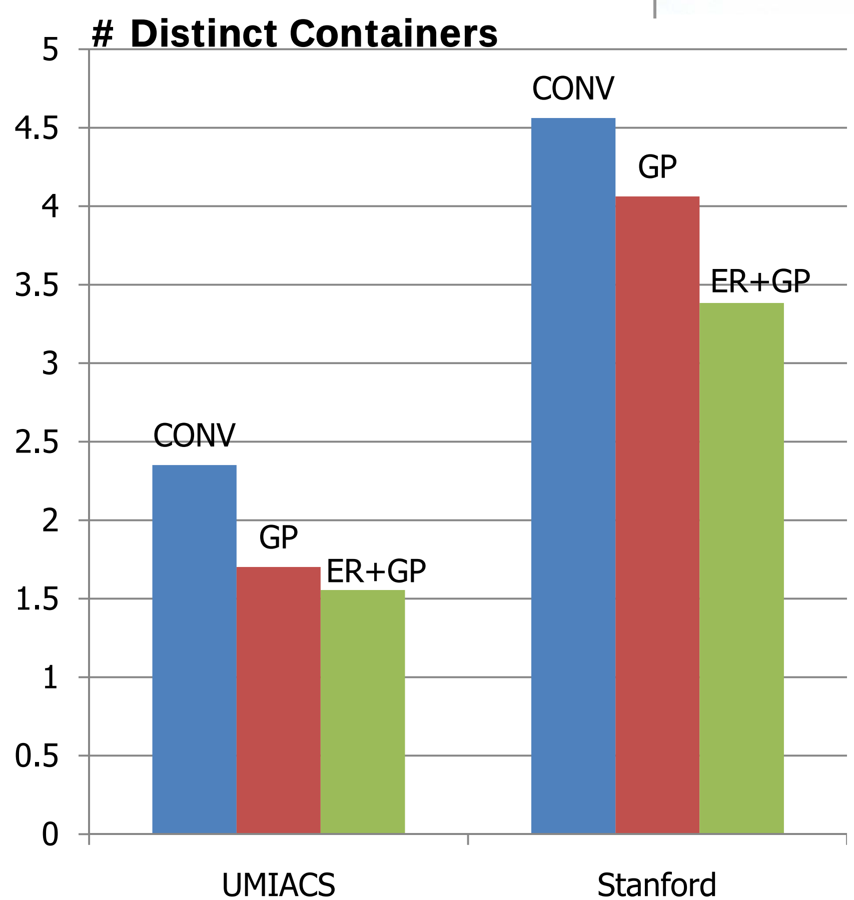
Simulation Results (Stanford)





Simulation Results





Conclusion

Conclusion

- We have shown that a graph partitioning scheme for organizing archive containers significantly reduces the number of containers that need to be accessed when a user browses through the archived web material.
- A graph analysis technique can improve this number even further.
- The overhead required by this technique is relatively small. On our 2 Ghz Intel Core 2 Duo processor, we could fully partition and compute EdgeRank of a large graph (the Stanford web graph that contains about 300,000 vertices, and 2.3 million edges) within minutes.
- Our simulation considers random access pattern on a single version of pages. However, the Web Graph model, and graph analysis technique can be extended to accommodate other access patterns on multiple versions.