Building a Scalable Web Server with Global Object Space Support on Heterogeneous Clusters

Ge Chen, Cho-Li Wang, Francis. C. M. Lau
Department Of Computer Science and Information Systems
The University of Hong Kong

The Need for Speed

- Internet user is growing very fast
  - According to United States Internet Council's report, regular Internet user has increased from less then 9,000,000 in 1993 to more than 300,000,000 in the summer of 2000, and is still growing fast
- Broadband becomes popular
  - According IDG's report, 57% of the workers in U.S access Internet via broadband in office. The figure will be more than 90% by 2005. Home broadband user will also increase from less than 9,000,000 now to over 55,000,000 by 2005
- HTTP requests account for larger portion of Internet traffic now
  - One study shows that HTTP activity has grown to account for 75%~80% of all Internet traffic

The Need for Speed

  - Growing user number
  - Faster connection speed
  - Increasing portion of HTTP requests accounts for all Internet traffic
- Require a more powerful Web server architecture

Cluster Based Solution

- Clustering is one of the promising solutions
- It is widely adopted by industry and academic projects
  - Google.com
  - AltaVista.com
  - Swala Project at UCSB
  - DC-Apache at the University of Arizona
  - More ...
- Previous researches mainly focus on:
  - Load Balancing [H. Bryhni et al. 2000]
  - Scalability [Trevor Schroeder et al. 2000]
  - High Availability [Guillaume Pierre et al. 2000]
- But not so much on:
  - Server-side caching in a cluster environment

Web Caching

- Web caching is one of the most effective ways to improve internet access quality
- Web caching is implemented at different levels of the Web hierarchy
  - client-side (WM.sers)archy

Cooperative Caching

- *Cooperative caching* is an enhancement of traditional caching mechanisms
- Cooperative caching has been used in Web proxy caching systems and distributed file systems
- Cooperative Web proxy caching is usually employed in wide area networks
  - Long network latency
  - Relative unstable connection
  - High communication cost
- It is impractical to adopt complicated caching algorithms in a WAN environment

Cooperative Caching

- File systems usually cache data at fixed-size blocks, while Web object are of various size
- File system access pattern usually exhibits strong temporal locality of references
- LRU or its derivatives are usually well-enough to maintain the blocks in the cache
- Cooperative caching in distributed file system usually has complicated and sophisticated consistency model

Global Object Space

- We propose a *Global Object Space (GOS)* in cluster based Web server
  - On top of the physical memory of all the cluster nodes
  - GOS is used to cache object content
  - All Web objects in system are visible and accessible to every node through GOS
- Clusters are tightly connected by high-speed LANs, communication cost is relatively small
- It is possible to maintain more accurate system-wide information about the cached objects

Hot Objects

- Web object access pattern exhibits strong *concentration*
  - Some study shows that around 10% of the distinct documents are responsible for 80-95% of all requests received by a Web server
- We use “hot objects” to refer to objects that are frequently requested in a short interval

Global Object Space Support

- Two tables are maintained in GOS at each node: *Global Object Table (GOT)* and *Local Object Table (LOT)*
  - *Home node* of an object refers to the node holds the persistent copy of that object
  - GOT keeps system wide information for all objects in system
    - Home node of an object
    - Location of cached copies of local objects
    - The approximated global access counter

Global Object Space Support

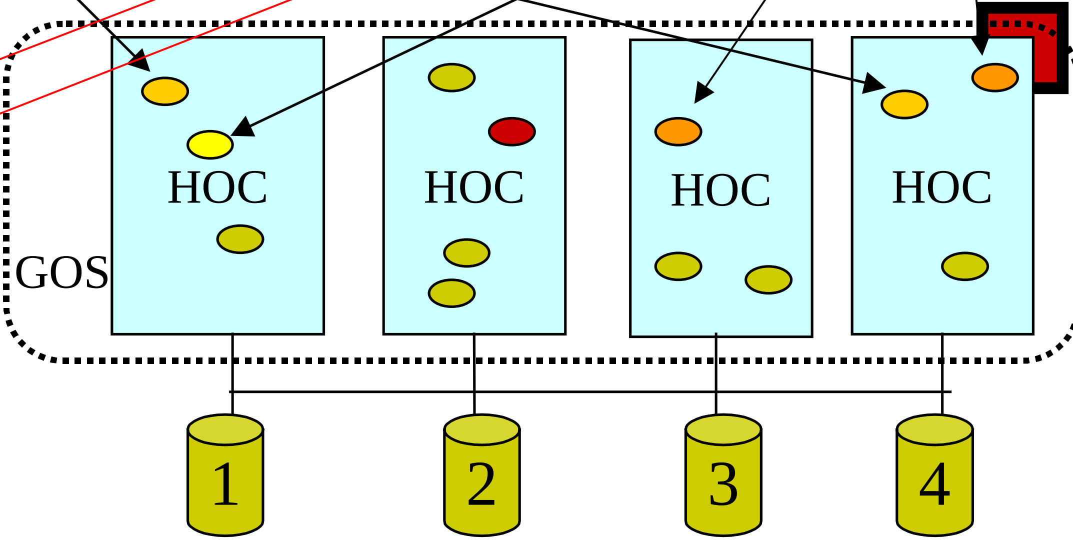
- LOT keeps access records for objects cached in current node's hot object cache
  - The approximated global access counter
  - Local access counter
  - Home node of the cached object

Global Object Space Support

GOT for Node 1			
Object URL/Partial URL	AGAC	HN	
/node3/dir31/doc3.html		3	
/node4/dir41		4	
/node1/dir11/doc1.html	105	1	
...	...	...	

GOT for Node 3			
Object URL	AGAC 45+200 = 245	HN	
/node3/dir31/doc3.html	50	3	
/node3/dir32/pic3.jpg	245	3	
...	...	...	

LOT for Node 1			
Object URL	LAC	AGAC	HN
/node3/dir32/pic3.jpg	0	245	3
/node1/dir11/doc1.html	24	105	1
...	...	...	



- AGAC: Approximated Global Access Counter
- LAC : Local Access Counter
- HN : Home Node

Global Object Space Support

- Approximate global access information is maintained for every object cached in GOS
  - Every object cached in GOS has a global access counter in the LOT of its home node
  - Every cached object copy is associated with a local access counter in its caching node
  - Local access counter is periodically sent back to objects' home node to maintain an approximate global access counter for every cached object
  - The approximate global object access counter will be fed back to the cached copy node after the global access counter updated

Cache Replacement Policy

- A LFU-based algorithm, LFU-Aging, is used as cache replacement algorithm
  - The global access counter is used to decide which object to be replaced
  - Aging effect applies to the global access counter
  - The cached objects are those most frequently accessed system wide, these are “hottest” ones

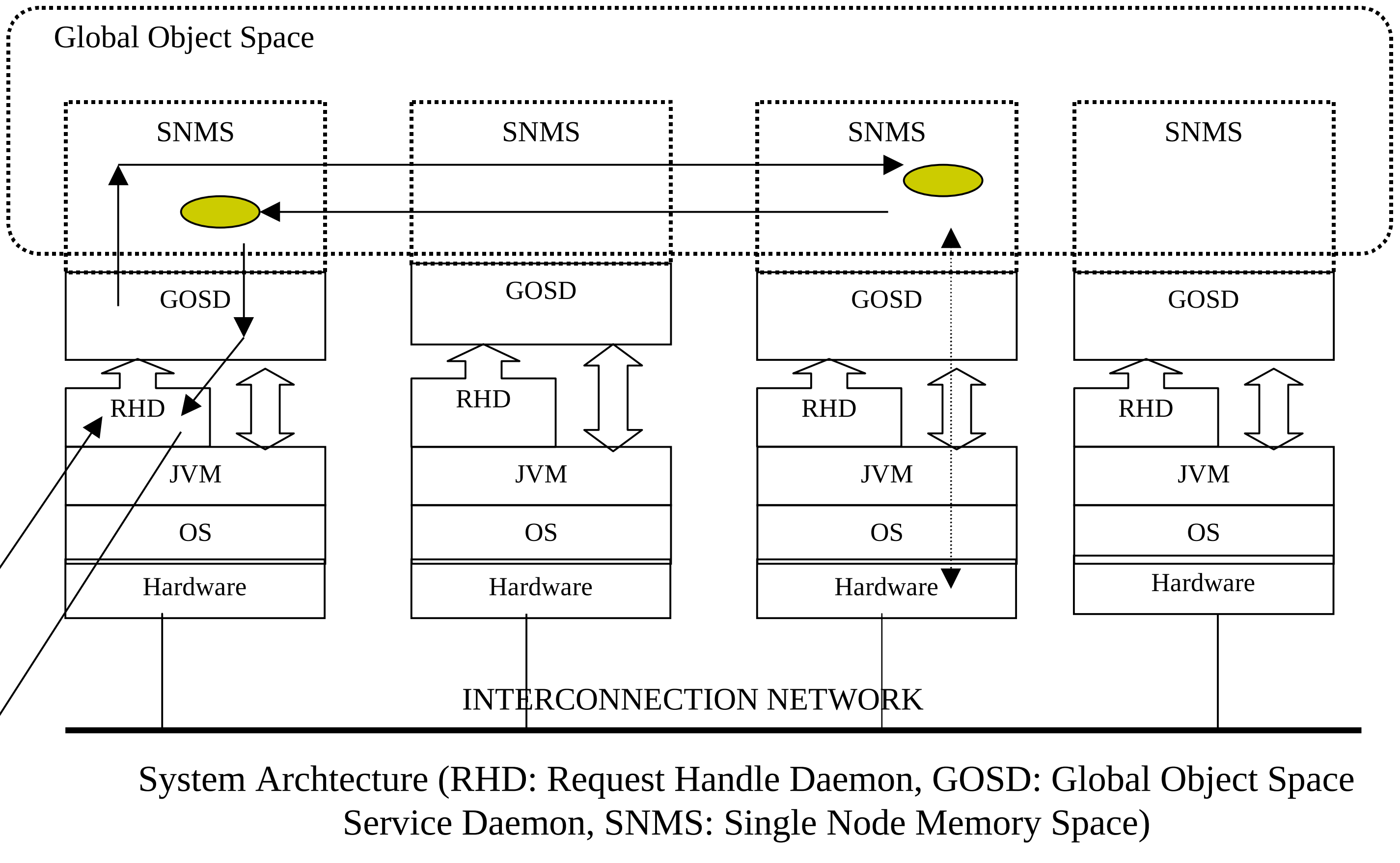
Global Object Space Support

- "Hot objects" may have duplicated copies because
  - The only criteria for whether caching an object or not is the global access counter returned or updated by the home node
  - Cached objects' life time is set according to HTTP specification's definition
  - Once an object becomes invalid before the predefined time, invalidation message is sent by home node to nodes caching the object
  - Duplicated copies can keep consistent by using above consistency protocol

Global Object Space Support

- Duplicated hot object copies will lead to
  - Improved local cache hit rate in the request servicing node by caching objects with largest approximate global access counter
  - Improved global hit rate by caching the "hottest objects" in global object space
  - Void excessive inter-node object fetching with higher local cache hit rate
  - Share the busty requests for hot objects to alleviate the workload of the home nodes holding many hot objects

Proposed System Architecture



System Archtecture (RHD: Request Handle Daemon, GOSD: Global Object Space Service Daemon, SNMS: Single Node Memory Space)

http://wsc.com/dir32/pic3.jpg

Proposed System Architecture

- RHD will accept all incoming HTTP request
- Parsed request is transferred to GOSD
- GOSD will try to find a cached copy in its own local memory
- If failed, GOSD will enquiry the home node for cached copy's node number
- After fetching an object, GOSD will try to cache the object in its local memory according the caching policy

Experiment Results

- Experiment Setup
  - 32-node PC cluster. Each node consists of a 733MHz Pentium III PC running Linux 2.2.4.
  - The nodes are connected with an 80-port Cisco Catalyst 2980G Fast Ethernet Switch.
  - 16 nodes acts as clients, and the rest as Web servers.
  - Each of the server nodes has 392MB physical memory installed

Experiment Results

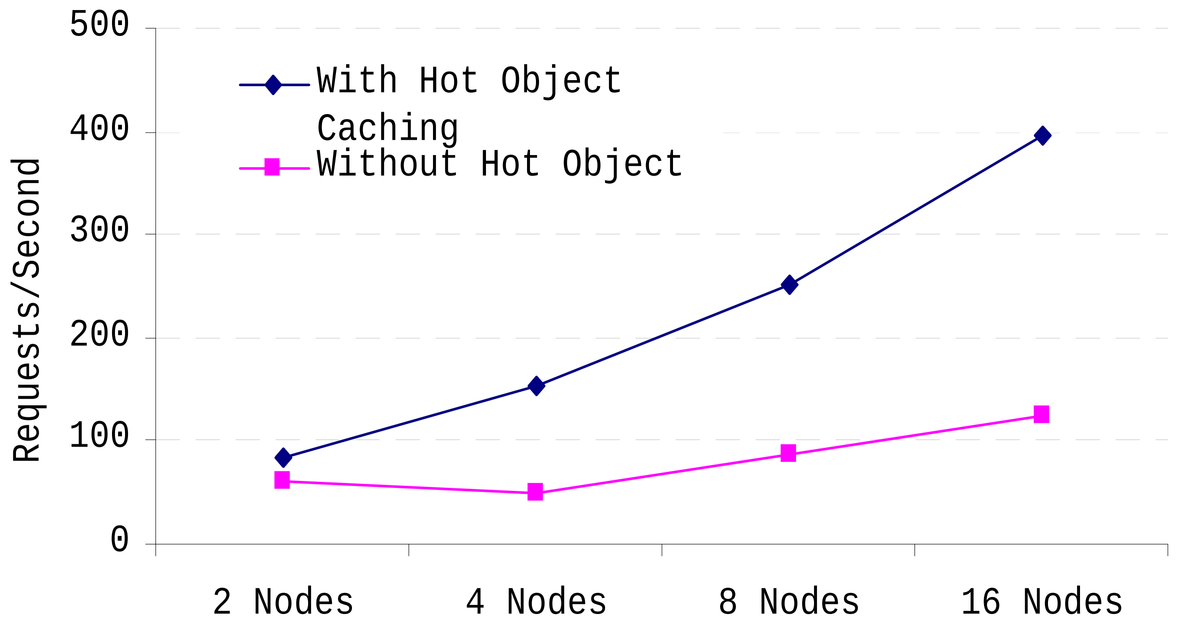
- A preliminary prototype system has been implemented by modifying the W3C's Jigsaw, version 2.0.5.
- The clients are modified version of httperf. It performs stress test on the designated Web server based on a collected Web server log from a well-known academic site.

Portability

- Pure-Java implementation makes our system portable to any system support Java
- Experiment was carried on a 4-node heterogeneous cluster consists of different hardware and software platform
  - One Celeron PC running Microsoft Windows 2000 Professional
  - One SMP PC with two Pentium Pro CPUs running RedHat Linux 6.1
  - Two PCs each with a Pentium II CPU running RedHat Linux 6.2

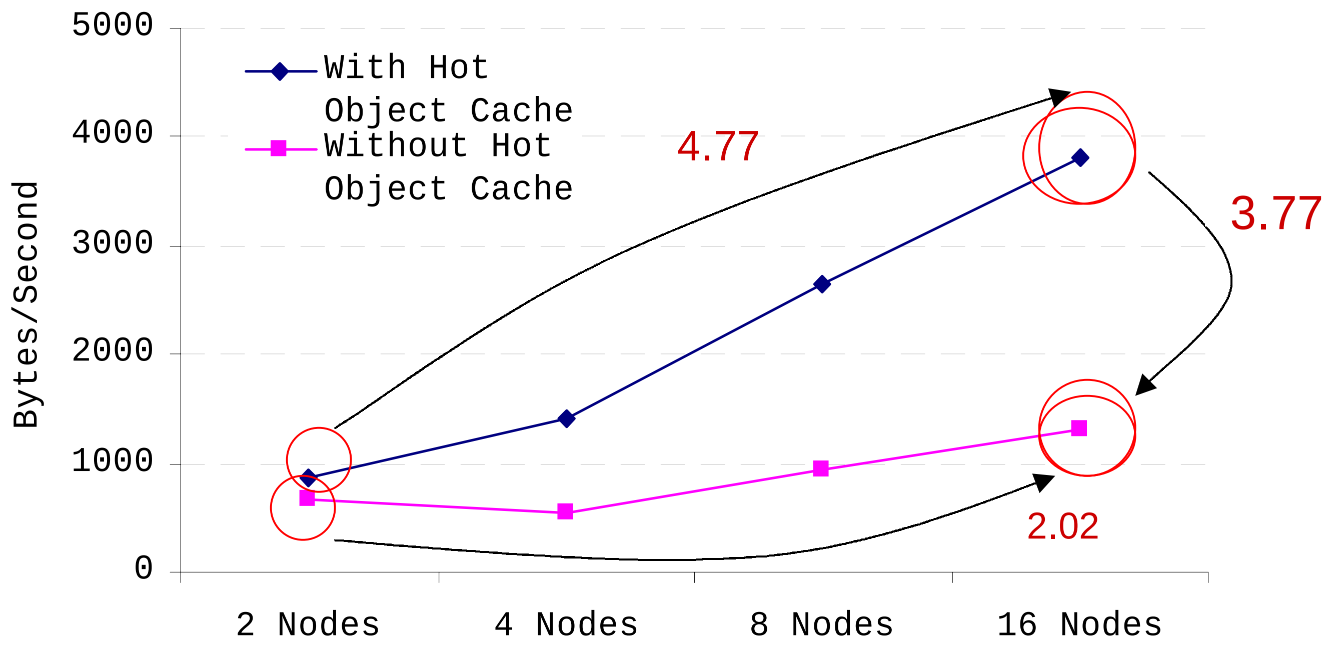
Experiment Results

- Effects of Scaling the Cluster Size



Experiment Results

- Effects of Scaling the Cluster Size

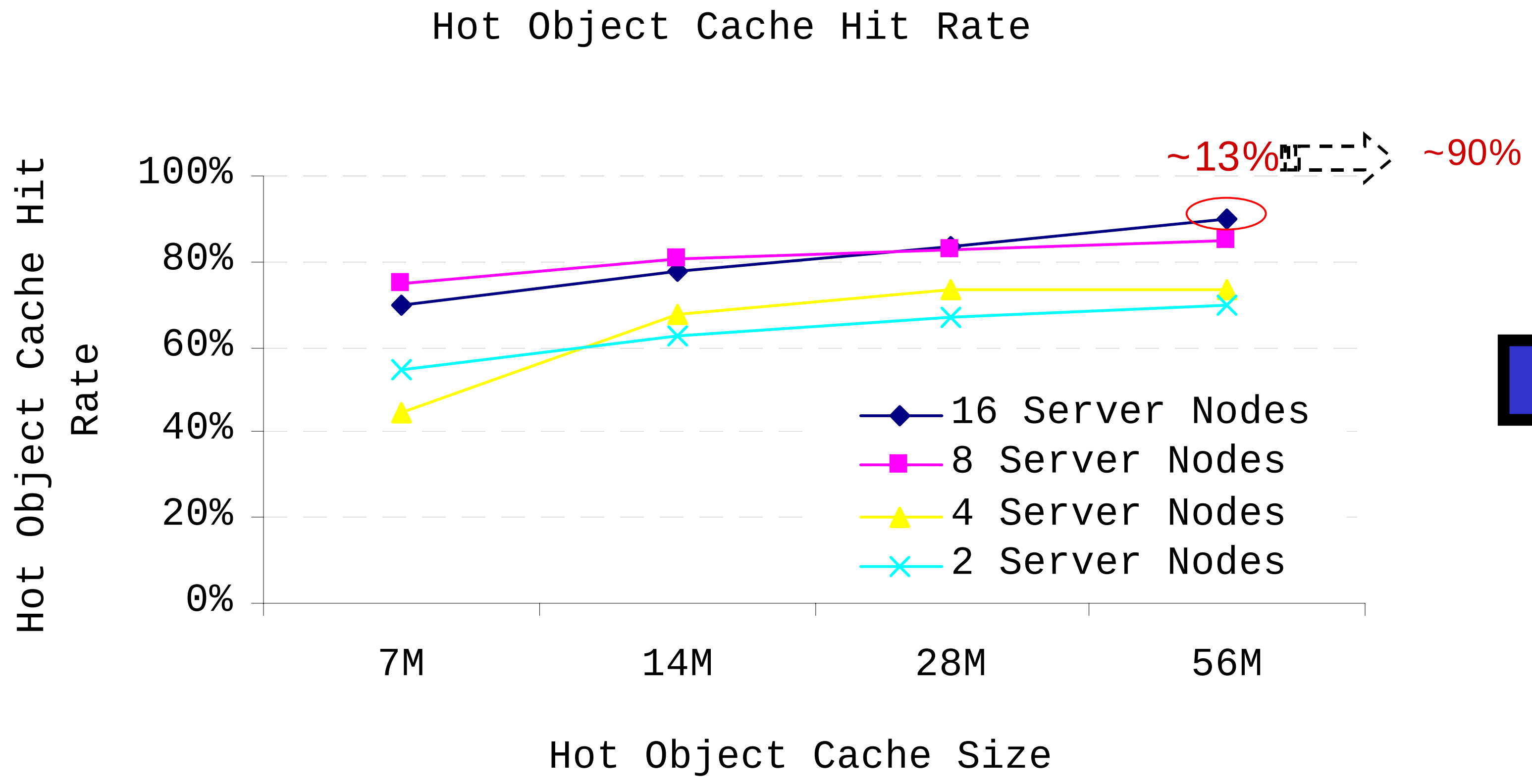


Experiment Results

- Effects of Scaling the Cluster Size
  - The performance with GOS enabled is almost 3.7 times of that without GOS in the sixteen-node case
  - Speedup is 4.77 when system scales from 2 nodes to 16 nodes with GOS support
  - Speedup is only 2.02 without GOS support
  - GOS has very positive impacts on the whole system performance and scalability.

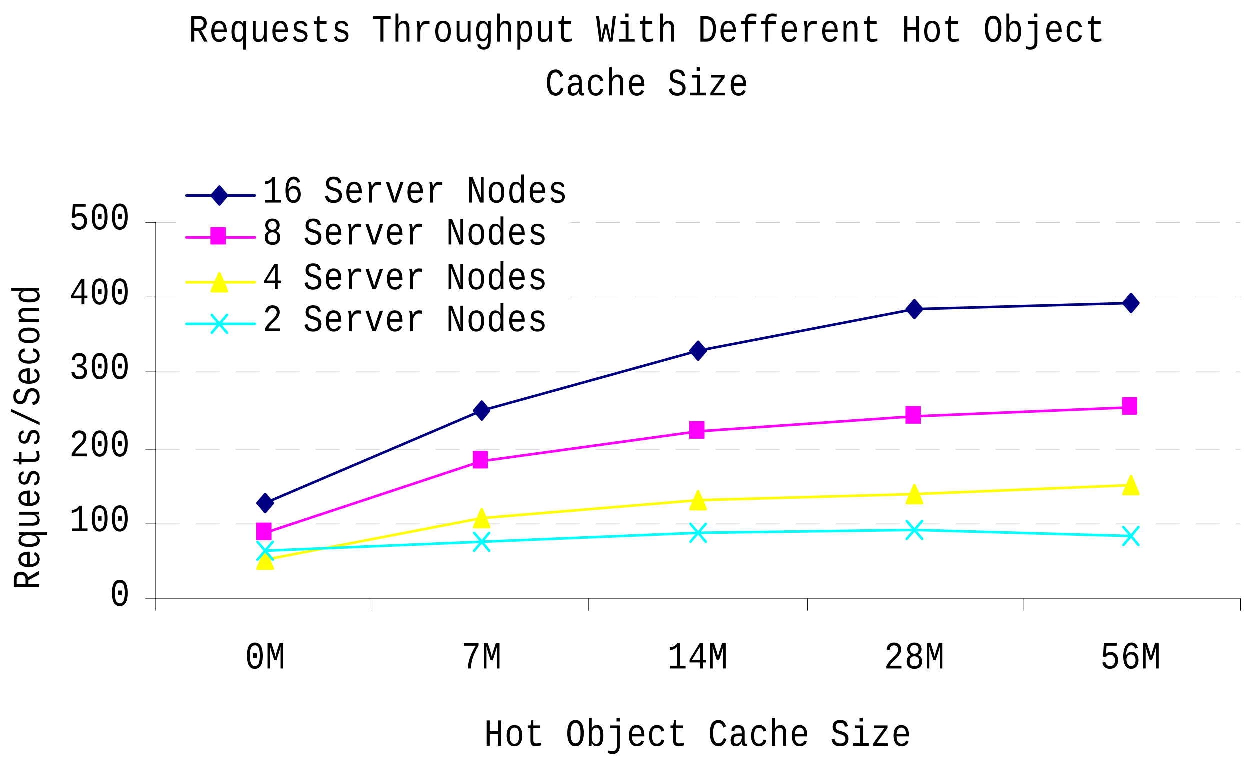
Experiment Results

- Effects of Scaling the Cache Size



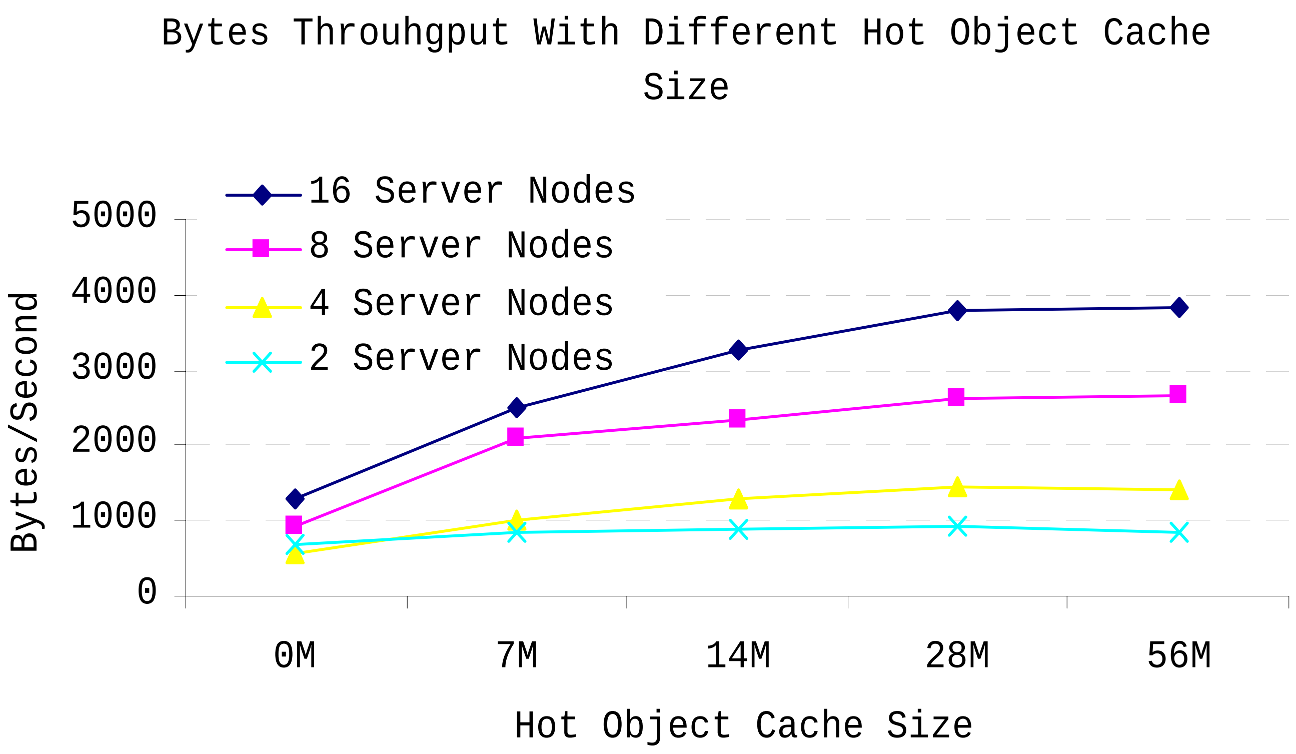
Experiment Results

- Effects of Scaling the Cache Size



Experiment Results

- Effects of Scaling the Cache Size

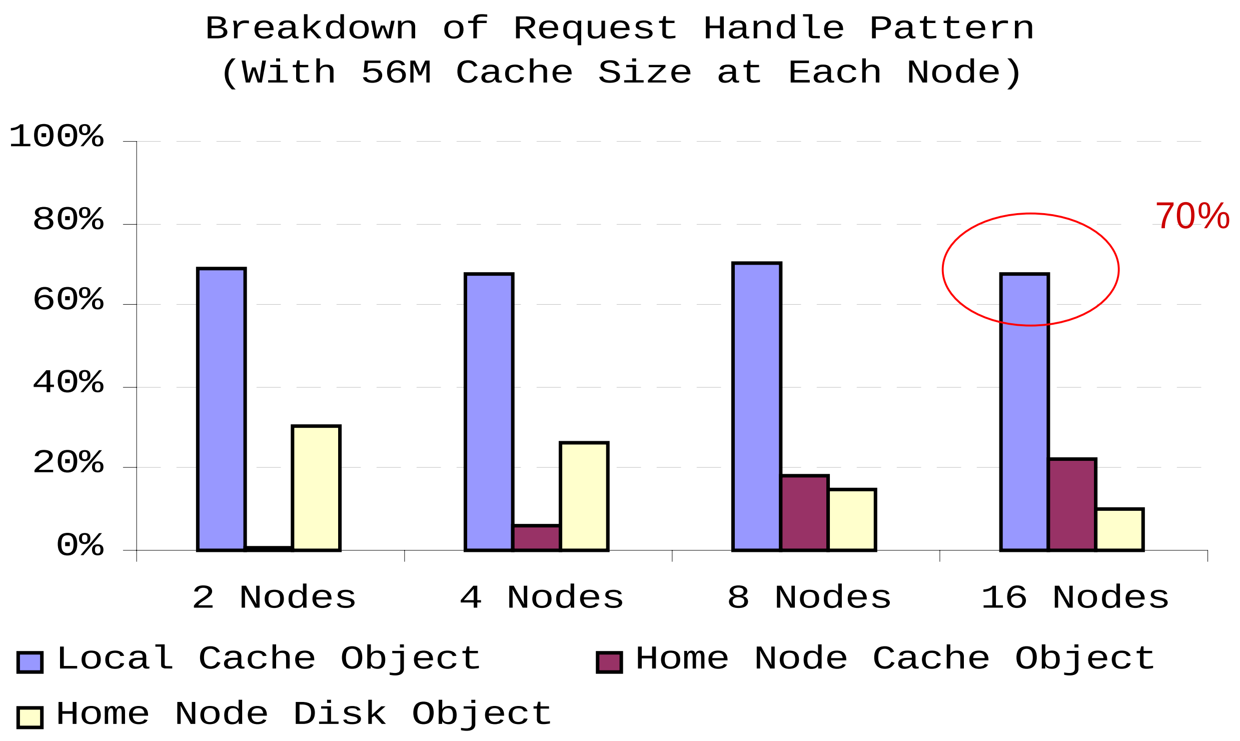


Experiment Results

- Effects of Scaling the Cache Size
  - The largest total cache size is about 13% of the data set size, the cache hit rate reaches around 90%
  - This confirms the early research observation that around 10% of the distinct objects account for 80%~95% of all the requests the server received
  - The approximated global LFU algorithm is effective
  - With a relatively small amount of memory in each node used for caching hot objects, we are able to obtain a high cache hit rate which increases the whole system’s performance considerably

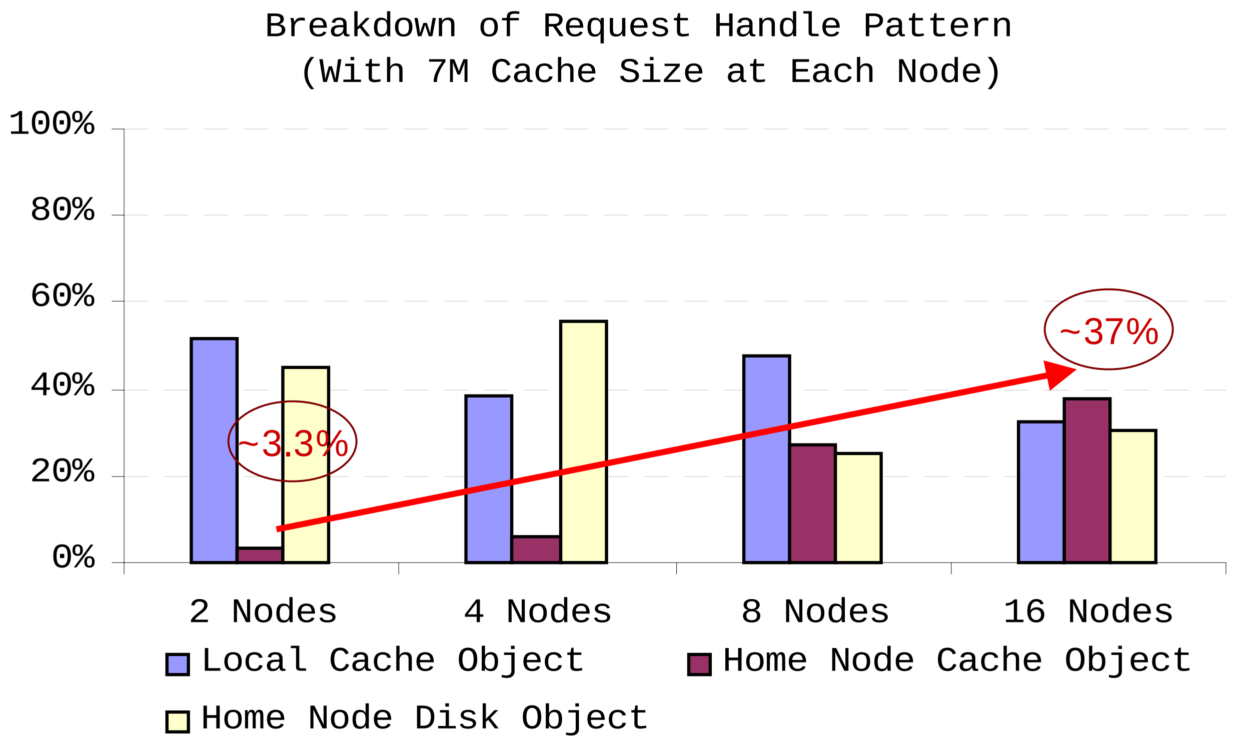
Experiment Results

- Requests Handle Pattern



Experiment Results

- Requests Handle Pattern



Experiment Results

- Requests Handle Pattern
  - Local cache hit rate can reach as high as 70%, this greatly reduces response time, therefore, the whole system throughput
  - It proves again that our approximated global LFU algorithm is very effective
  - As the number of cluster nodes increases, the home node cache objects account for a larger portion of the total global cache hit rate
  - It shows the advantage of adopting the cooperative caching technique in our system for improving the performance by reducing the disk operations

Related Works

- DC-Apache (DCWS) [Scott M. Baker et al.: 1998]
  - Dynamically manipulate the hyperlinks stored within the Web documents
  - Dynamically migrates documents to cooperative Web servers which are dedicated server nodes for sharing the load of the home server
- Localtion Aware Request Distribution in CS, Rice University
  - Uses a front-end server as request dispatcher
  - Request from a new client will be dispatched to one of the backend server nodes according to the LARD policy

Related Works

- KAIST's work [Woo Hyun Ahn et al.: 2000]
  - Cluster-based file system with cooperative caching designed for supporting parallel Web server systems
  - Based on some hint-based file system cooperative cache
  - Positive results are obtained from the simulation experiments indicating good load balancing and reduced disk access rates

Conclusions

- Use of cluster wide physical memory as object cache can lead to improved performance and scalability of Web server systems
- With relatively small amount of memory dedicated for object content caching, we are able to achieve a high hit rate
- Cooperative caching can further improve the global hit rate

Conclusions

- It is possible to provide extension to dynamic content caching by implementing more sophisticated object consistency protocol within the global object space
- Although our approach is proposed for Web server systems, the proposed caching mechanism is also suitable for building cluster based proxy servers