Towards Building the OceanStore Web Cache

Patrick R. Eaton
University of California, Berkeley
`eaton@cs.berkeley.edu`

June 10, 2002

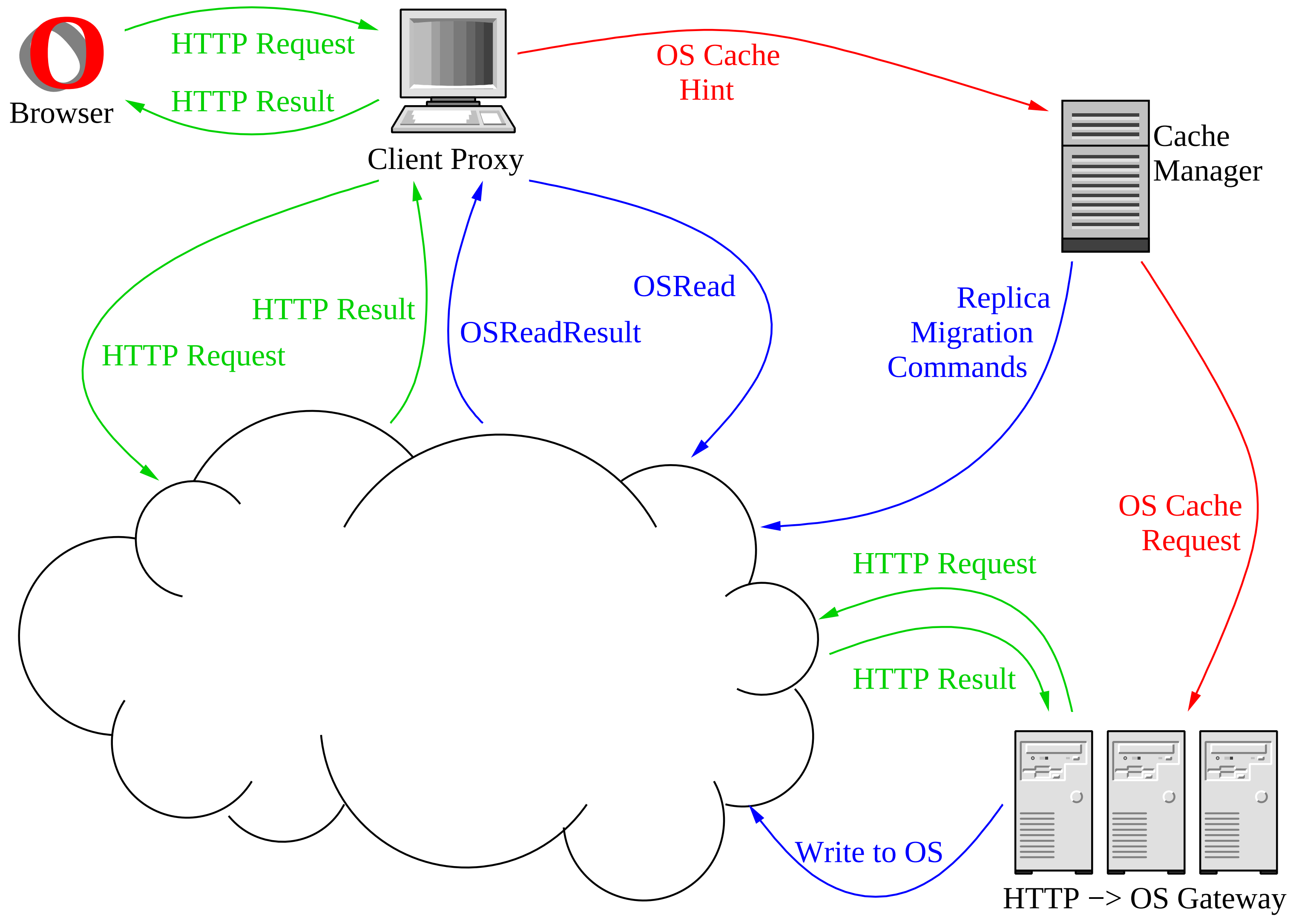
Motivation

- Traditional hierarchical web caching architectures require much maintenance and human configuration.
- We have developed a web cache architecture which exploits the features of OceanStore to be self-configuring/managing/maintaining.
  - uses Tapestry to allow cache nodes to enter and leave the network without impacting other caches
  - uses Tapestry to locate objects in the network without explicit knowledge of other caches
  - uses excess resources in the network to cache more content
- What is the cost in performance of this new architecture?

Components of the OceanStore Web Caching Architecture

- Client proxy.
  - translates a user's web requests to check the OceanStore web cache
  - runs on same machine as user's web browser
- HTTP to OceanStore gateway.
  - convert web content into OceanStore documents
  - hosted by regional cache provider
- Cache managers.
  - work greedily to provide best level of service to clients in the local area
  - run locally by department or organization

The OceanStore Web Cache Architecture



Client Proxy

- Check URL for hints on cacheability.
  - cookies, CGI scripts, embedded variables...
  - previously-accessed URL that was found to be uncacheable
- If uncacheable, forward the request to the origin server.
- If cacheable, translate the request and forward to the web cache.
- If the web cache responds, translate and server to the client; otherwise, forward to the origin server.
- If time to retrieve the document was unreasonable, send a service request to a nearby cache manager.

Gateway

- Distributed throughout infrastructure.
- Published in Tapestry by a well-known GUID.
- Accept requests for documents missing from the cache.
- Retrieve document from the origin server.
- If cacheable, write the content into the web cache.
  - create a new object and write the content into it
  - update the object storing the content
- Key management isolated to this component.

Cache Manager

- The introspective agent in the web cache architecture.
- Distributed throughout the infrastructure.
- Published in Tapestry by a well-known GUID.
- Respond to user access patterns by directing the number and location of replicas.
- Most useful to nearby clients.
  - run by organizations for the benefit of their users

Scalability and Maintainability

- Tapestry allows nodes to enter and leave the network without notice.
- Tapestry allows us to locate service providers.
- No hierarchy or group configuration/maintenance.
- Efficient use of excess resources in the network.
- No network “hot-spots”.
- Greater aggregate read bandwidth.

Implementation

- Built the proxy, gateway, and cache manager.
- The proxy and gateway are fully implemented as described above.
- The cache manager is a very limited implementation.
  - forwards cache misses to the gateway
  - on a cache miss, creates on replica of the document on a random node in the system
  - no further replica management

Experimental Setup

- Ninety-eight (98) OceanStore nodes placed randomly on a 496-node transit stub network.
  - ˜150 ms inter-domain latency
  - 10-50 ms intra-domain latency
  - these latencies are 2x what we have observed
- Use Tapestry base of two bits
  - results in location lookups of up to seven Tapestry hops longs
- Run on 42-node ROC cluster
  - 8 OceanStore nodes per cluster node
  - dual 1.0 GHz Pentium III CPUs
  - 1.6 GB ECC PC133 SDRAM
  - two 36 GB IBM hard drives
  - gigabit ethernet
- Web server is run on the management node of the cluster.

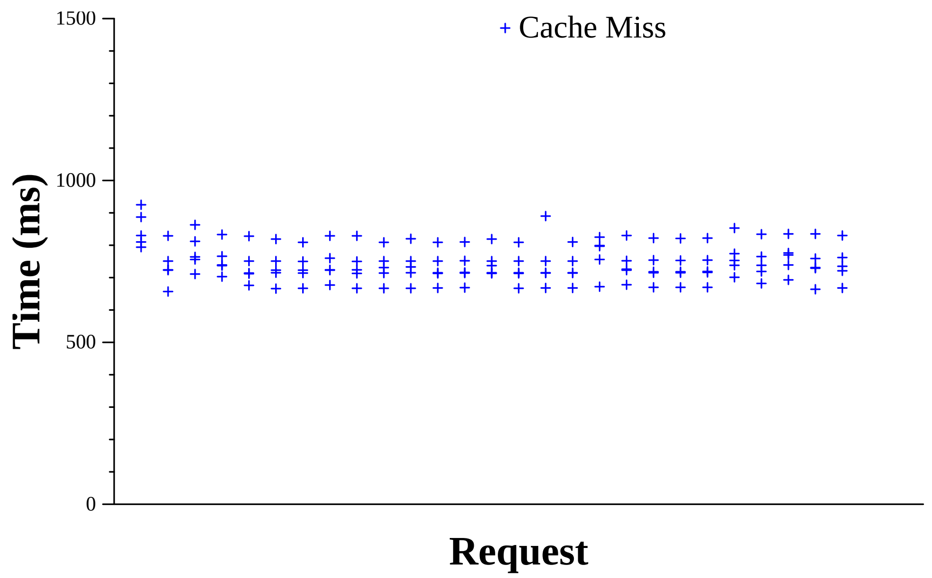
Internet Cache Protocol (ICP)

- A simple, light-weight, hierarchical caching scheme.
- Clients are configured to send all requests to a proxy.
- A proxy responds from a local cache or queries a number of peer caches for the content.
- If no peer has the document, the proxy forwards the request to the next level in the hierarchy.
- Cost of maintaining a set of peer caches is high.
- The foundation of many products.
  - Squid, Cisco Cache Engine, Novell Internet Cache System, Microsoft Proxy

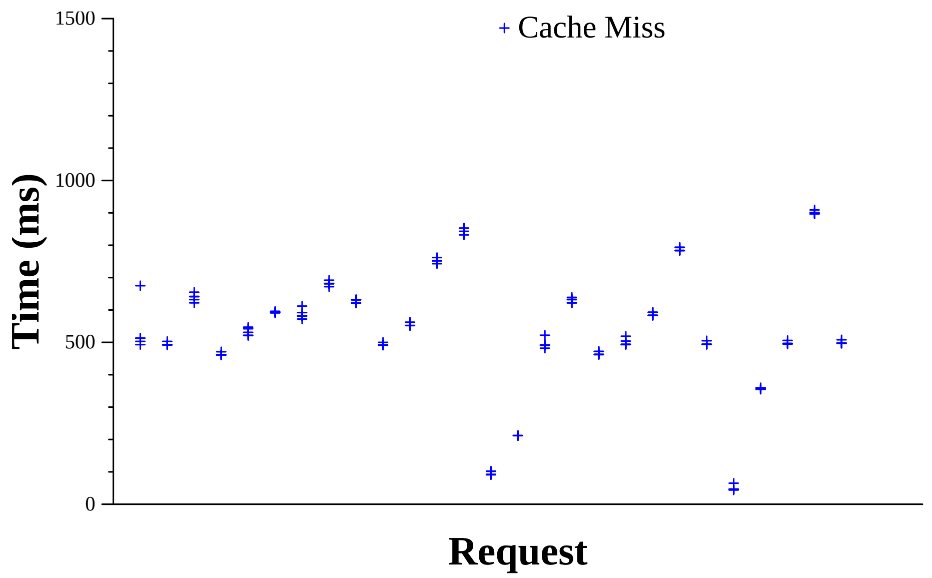
Cache Latency

- Measure the latency of a single request.
- Cache miss.
  - document is not cached on any node
  - retrieve document from origin server after lookup fails
- Local hit.
  - document is cached locally
  - can return document immediately
- Remote hit.
  - document is not cached locally but is cached on some node
  - must find node with content cached and retrieve document
- Key difference between caches.
  - OceanStore searches other caches through a series of serial Tapestry hops
  - ICP searches other caches through a parallel multicast

Cache Latency: Cache Miss



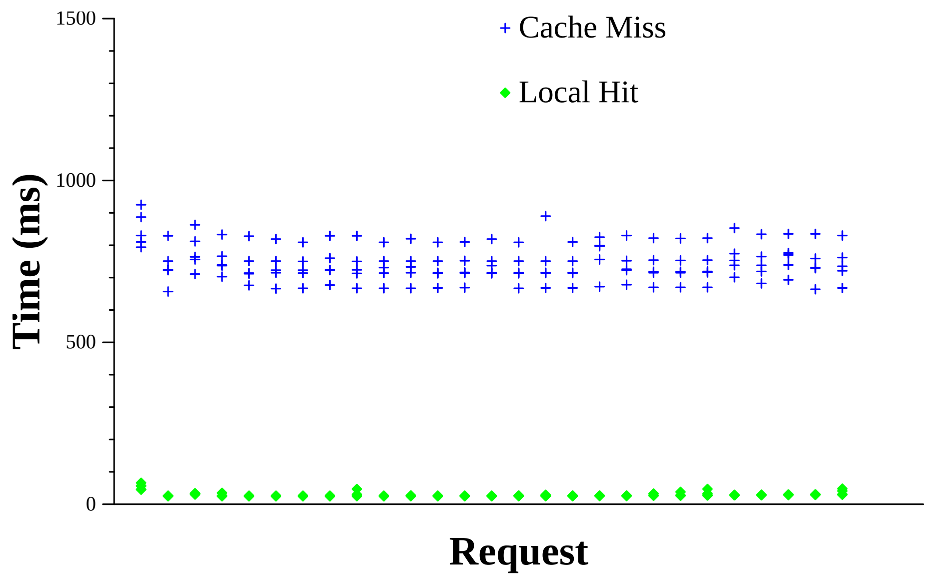
Latency of ICP Cache



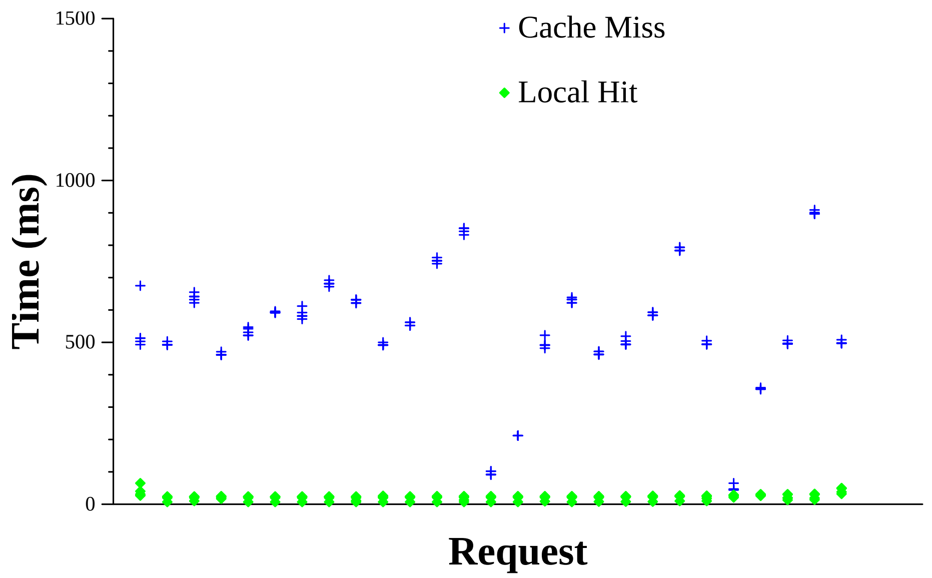
Latency of OceanStore Web Cache

- ICP cache waits to receive all nacks before requesting the document from the origin server.
- OceanStore cache requests document from origin server when Tapestry resolves that the document is not published in the network.

Cache Latency: Local Hit



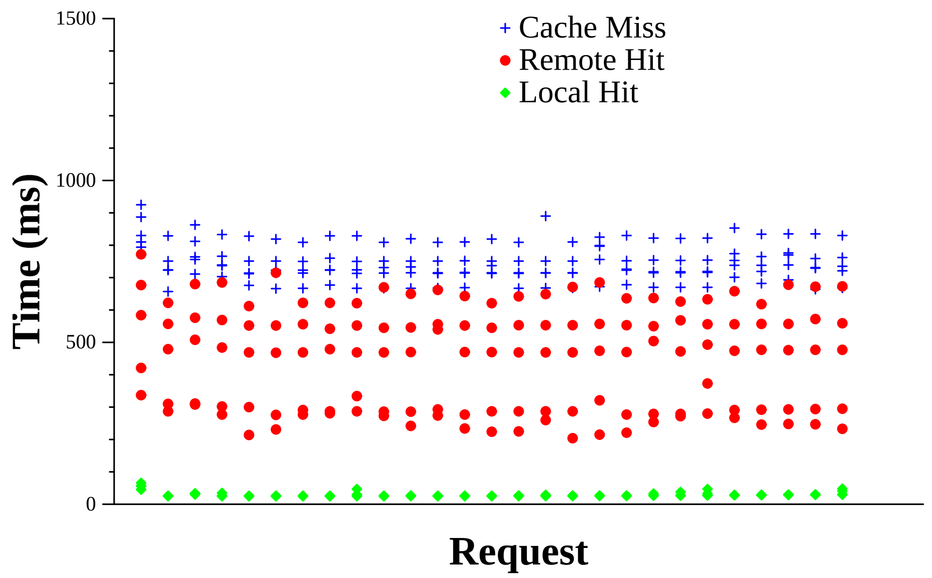
Latency of ICP Cache



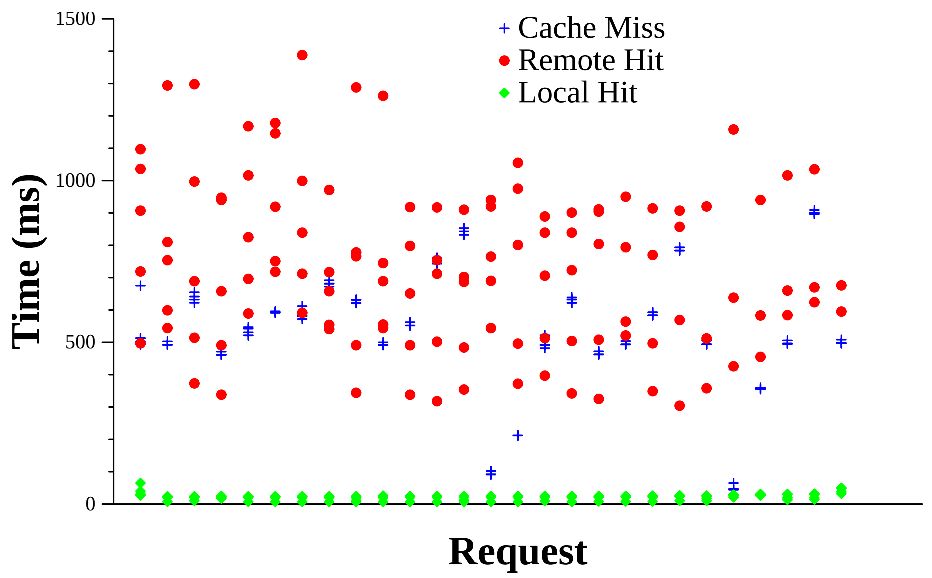
Latency of OceanStore Web Cache

- Both caches respond very quickly when document is cached locally.
- OceanStore cache actually serves close content twice as fast as the ICP cache (20 ms versus 35 ms).
  - OceanStore cache can move content to the requesting client
  - ICP cache can only move content to the proxy of the requesting client

Cache Latency: Remote Hit - The Bad News



Latency of ICP Cache



Latency of OceanStore Web Cache

- Can observe the effect of Tapestry's hop-by-hop routing.
  - highlights the importance of managing replicas to ensure content is close to consumers
- OceanStore cache can actually serve content faster when it is nearby.

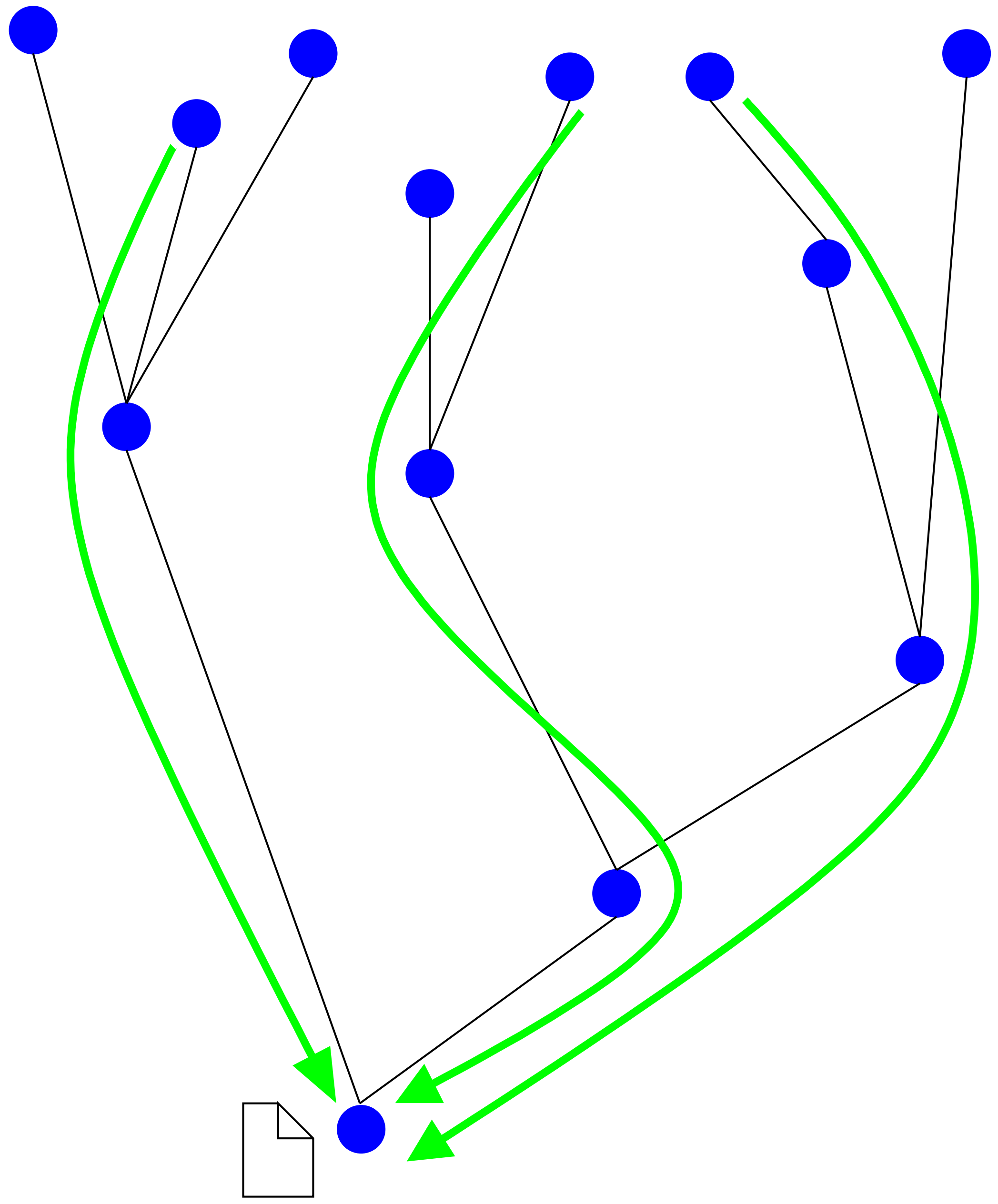
Inspiration for Replica Placement Strategy

In a system of tributaries, streams combine at a conflu-ence to form larger streams. Drops of water are routed from tiny brooks through larger streams to lakes, seas, and oceans.

“Tributaries” by Rob Gonsalves.

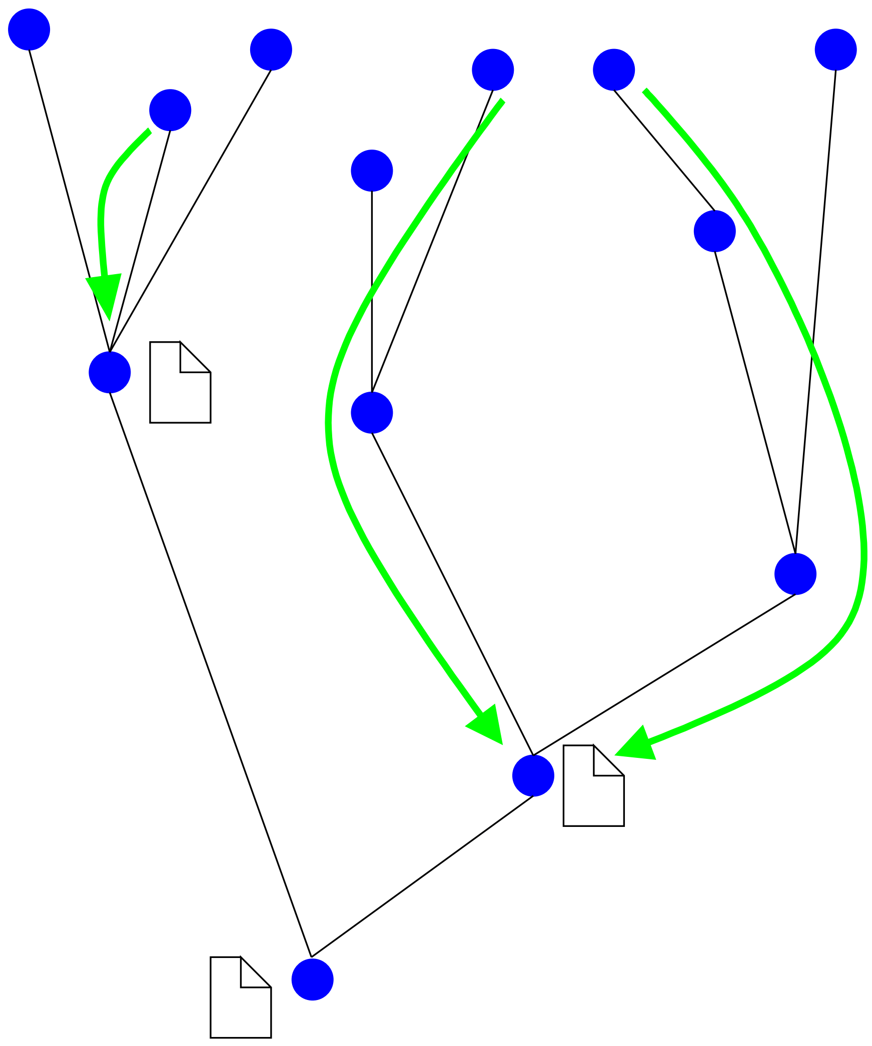
Inspiration for Replica Placement Strategy

In Tapestry, object location paths combine at Tapestry nodes. Location requests are routed from the edges of the network toward the object's Tapestry root.



Replica Placement Strategy

- Idea: Place replicas at the "confluence" of location paths.
- All clients "upstream" of the replica will benefit from it.



Ongoing Work

- Implement replica management in the cache managers.
- Explore use of Tapestry “time-outs” to reduce the cost of remote hits.
- Measure the effect of using idle resources in the network.
- Find appropriate workloads/load generators for measuring the system.

Conclusions

- The performance of individual components is adequate.
- The key to good aggregate performance is effective replica management.
- Next steps: improve the responsiveness of the cache managers.