

**On Differentiating
Distribution Strategies
for
Internet Applications**

Maarten van Steen

Guillaume Pierre, Ihor Kuz, Andrew Tanenbaum

Vrije Universiteit Amsterdam, Faculty of Science

Dept. Mathematics and Computer Science

Tel: (020) 444 7784

steen@cs.vu.nl

<http://www.cs.vu.nl/~steen>

<http://www.cs.vu.nl/globe>

Introduction (1/2)

Our research: building a wide-area distributed system called **Globe**:

- Capable of supporting a billion users, each having 1,000 objects
- Spread across the Internet
- Flexible support for a myriad of distribution strategies:
 - Objects decide how they should be replicated
 - Objects decide how their state should be distributed
 - Objects incorporate their own solution to fault tolerance
 - Objects decide on when and how they migrate
 - ...

Problem: How do we find out what kind of behavior we can expect from objects? Needed to determine actual strategies.

Introduction (2/2)

Before we actually address problem of identifying behavior patterns, we need to ask ourselves:

Question: Is it really necessary to differentiate distribution strategies?

And if so,

Question: How do we differentiate distribution strategies?

Both questions are addressed in this presentation.

Web Replication

Issue: Consider a collection of Web documents on a Web server. Caching and replication helps to improve performance:

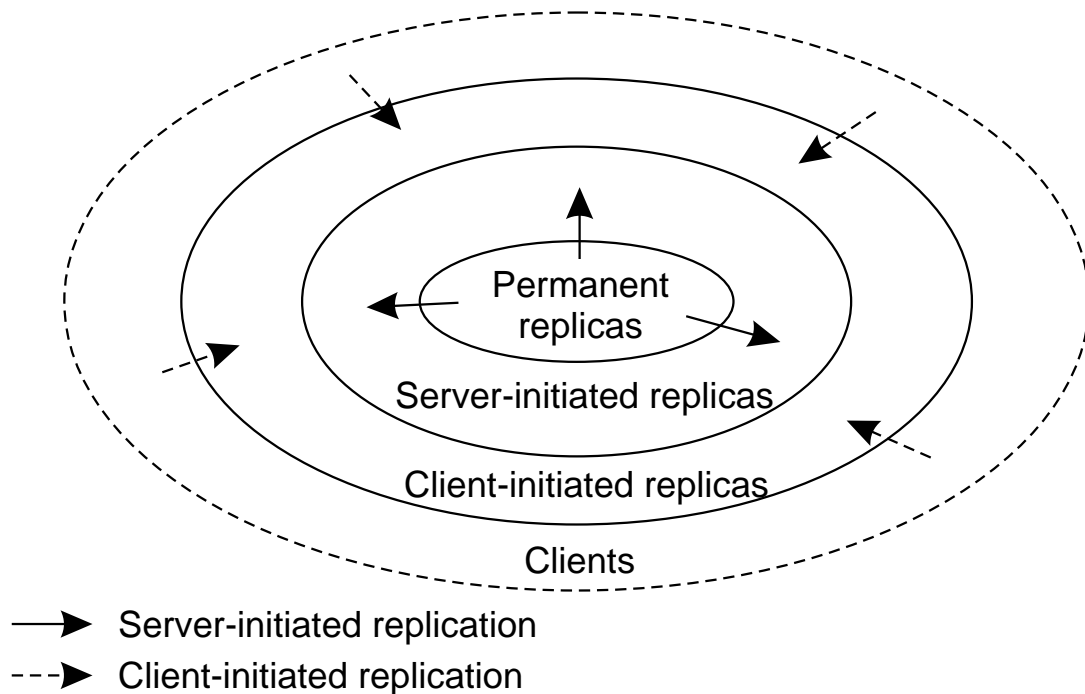
- Copies close to a client reduce access times
- Having copies may reduce consumed network bandwidth
- Balancing the load across several servers is good for the servers

Current Web: client-side caching, mirroring Web sites, and replication through Content Delivery Networks.

Problem: Caching and replication strategy is essentially the same for all pages, there is generally no relationship to the usage or content of a page.

Question: Are there actually that many different replication strategies?

A Model for Replication



Permanent replicas: Set of replicas that form the core of a (replicated) Web document

Server-initiated replicas: Replicas created by a permanent replica

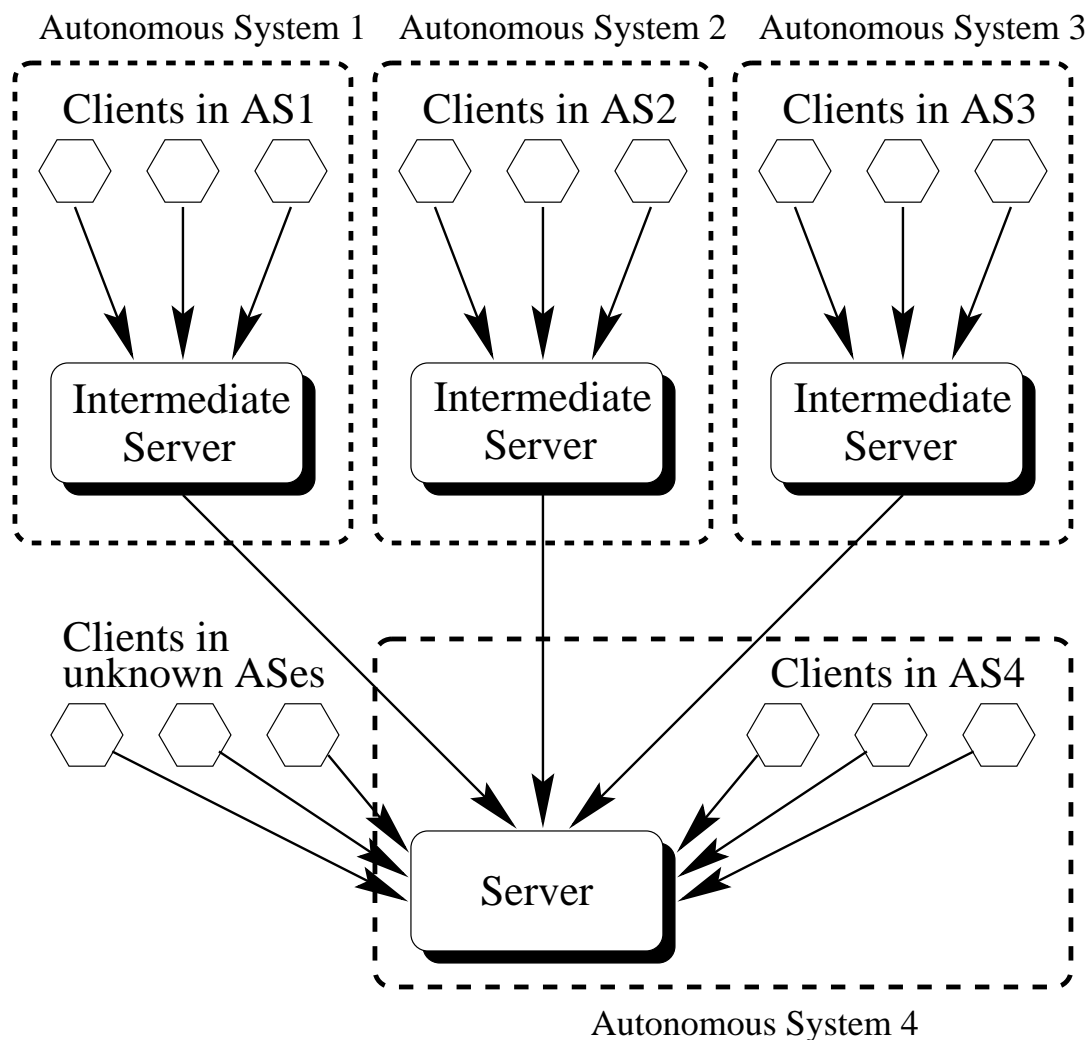
Client-initiated replicas: Caches created on behalf of clients

Differentiating Replication Strategies

Parameter	Values	Meaning
Change distribution	<ul style="list-style-type: none"> - notification - full state - state differences - operation 	Describes how changes between replicas are distributed: is only a notification (or invalidation) sent telling that an update is needed, is the full state sent, or only differences, or is the operation sent that is to be carried out to update the receiver's state.
Replica responsiveness	<ul style="list-style-type: none"> - immediate - lazy (e.g., periodic) - passive 	Describes how quickly a replica reacts when it notices it is no longer consistent with the other replicas. A passive replica will do nothing.
Replica reaction	<ul style="list-style-type: none"> - pull - push 	Describes what a (non passive) replica will do when it notices it is inconsistent with the other replicas. It will either push or request update messages.
Write set	<ul style="list-style-type: none"> - single - multiple 	This parameter gives the number of writers that may simultaneously access the Web document.
Coherence group	<ul style="list-style-type: none"> - permanent only - permanent and server-initiated - all layers 	Describes who implements the document's coherence model: permanent and/or server-initiated replicas.

Experiment (1/2)

Research question: Does it make sense to distribute each Web page according to its own “best” strategy, instead of applying a single, overall distribution strategy to all Web pages?



Experiment (2/2)

- We collected traces on requests *and* updates for all Web pages on our local server.
- For each request, we checked:
 - From which autonomous system it came
 - What the average delay was to that client
 - What the average bandwidth was to the client's AS (randomly taking 5 clients from that AS)
- Pages that were requested less than 10 times were removed from the experiment.
- We replayed the trace file for many different system configurations, and many different distribution scenarios.

Number of documents	17,368
Number of requests	2,118,572
Number of updates	9,143
Number of unique clients	107,386
Number of origin ASes	2,785

Caching Strategies

Note: Intermediate servers are configured as proxy caches.

No caching: clients contact the server directly.

Check: when a cache hit occurs, check consistency with server.

Alex: validity of a cached document depends on its last modification time: a recently modified cached document is removed from the cache sooner than a document that last modified a long time ago.

AlexCheck: same as Alex but documents are not removed, but first checked for consistency at the server (may save bandwidth).

CacheInv: server keeps track of where a document is cached, and sends an invalidation message when the document is modified.

Replication Strategies

Note: Intermediate servers are configured as replication servers and their copies are updated by the main server when the document is changed.

Strategy: take a look at the ASs where most requests come from, and place a replication server at the top ones:

Top ASs: 53% of all requests came from 10 ASs;
62% came from 25 ASs, and
71% came from 50 ASs.

Repl10: place a replication server at the 10 ASs from which most requests came.

Repl25: analogous for the 25 top ASs.

Repl50: analogous for the 50 top ASs.

Trace Results

Measurements: Calculate total delay, total number of inconsistent documents that clients read, and total consumed bandwidth at the server.

	Delay (sec.)	Incons. (#docs)	Bandwidth (Mbytes)
No repl	788553	5	44960
Check	824985	5	24164
Alex	346959	5218	24079
AlexCheck	347749	4828	23792
CacheInv	337144	5	23736
Repl10	638499	6	44643
Repl25	522015	6	49216
Repl50	438985	7	56885
Repl50+Alex	243122	971	48058
Repl50+AlexCheck	243410	946	47989

Big question: Can we assign a distribution strategy to each document, such that the minimal values are reached?

Assigning a Strategy (1/2)

Problem: What makes a strategy good is determined by (so far unspecified) objectives. For example, do we minimize on delay or minimize on bandwidth?

Solution: Let D be the set of documents, and S the set of different strategies. Let m_i denote a metric that is used to evaluate a strategy s (e.g., delay, bandwidth, etc.), and $m_i(d, s)$ the value computed for document d when it follows strategy s .

Assign a weight w_i to each metric, and calculate a (dimensionless) score $\sigma_{\mathbf{w}}(d, s)$ to each pair (d, s) :

$$\sigma_{\mathbf{w}}(d, s) = \langle w_1 \cdot m_1(d, s), \dots, w_k \cdot m_k(d, s) \rangle$$

with $\mathbf{w} = \langle w_1, \dots, w_k \rangle$ and k the number of different metrics.

Assigning a Strategy (2/2)

Arrangement: a set $A = \{(d, s) | d \in D, s \in S\}$. The score of an arrangement is defined as:

$$\sigma_{\mathbf{w}}(A) = \left\langle \sum_{(d,s) \in A} w_1 \cdot m_1(d, s), \dots, \sum_{(d,s) \in A} w_k \cdot m_k(d, s) \right\rangle$$

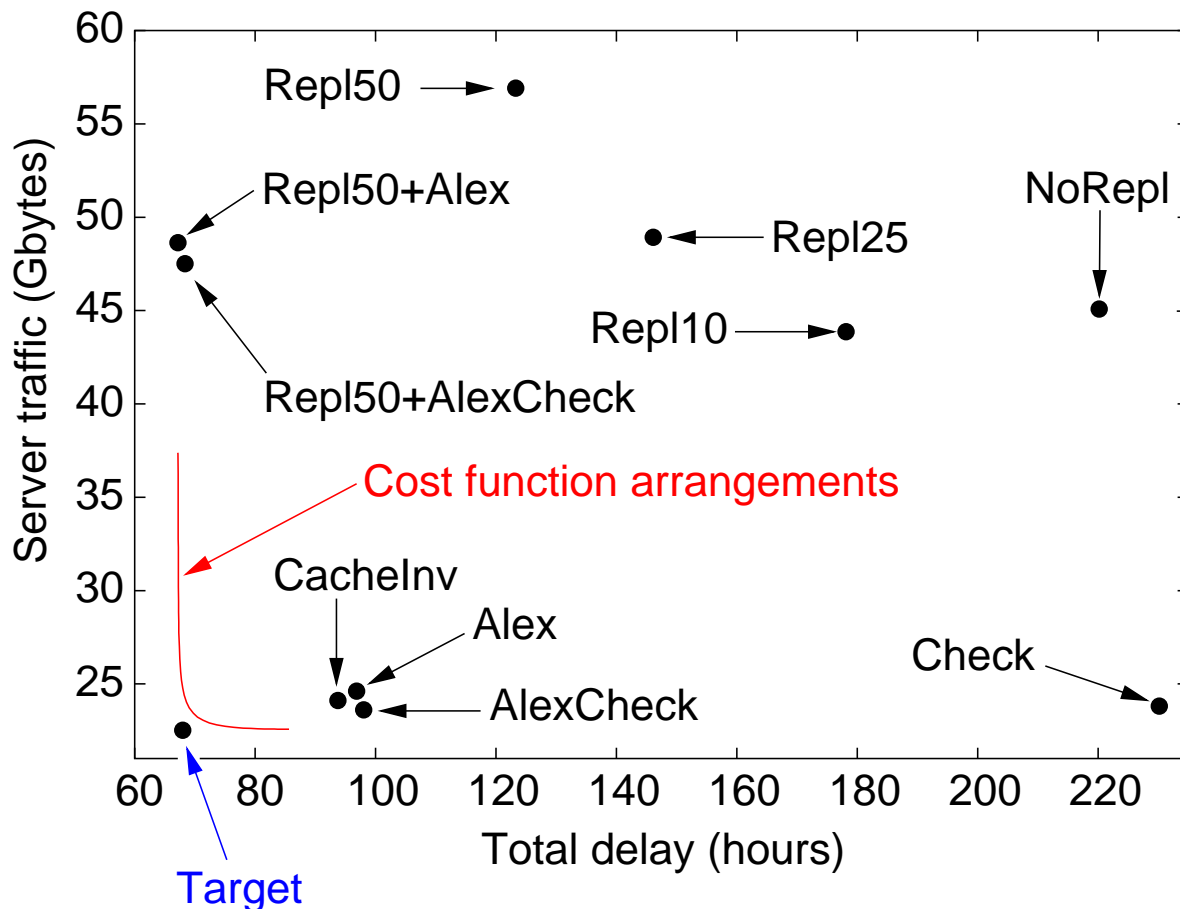
For our experiment, we insisted on having as few inconsistent copies as possible, leaving only delay and bandwidth as metrics.

For different combinations of w_1 and w_2 , we calculated a **cost** $G_{\mathbf{w}}(d, s)$:

$$G_{\mathbf{w}}(d, s) = w_1 \cdot \text{Delay}(d, s) + w_2 \cdot \text{Bandwidth}(d, s)$$

The strategy giving the minimal $G_{\mathbf{w}}(d, s)$ for a given \mathbf{w} and d was assigned to d .

Comparing Strategies (1/2)



Target: Best that can ever be achieved, i.e., minimal bandwidth and minimal delay.

Observation: Our cost function is doing better than any one-size-fits-all strategy \Rightarrow it makes sense to differentiate distribution strategies per document.

Comparing Strategies (2/2)

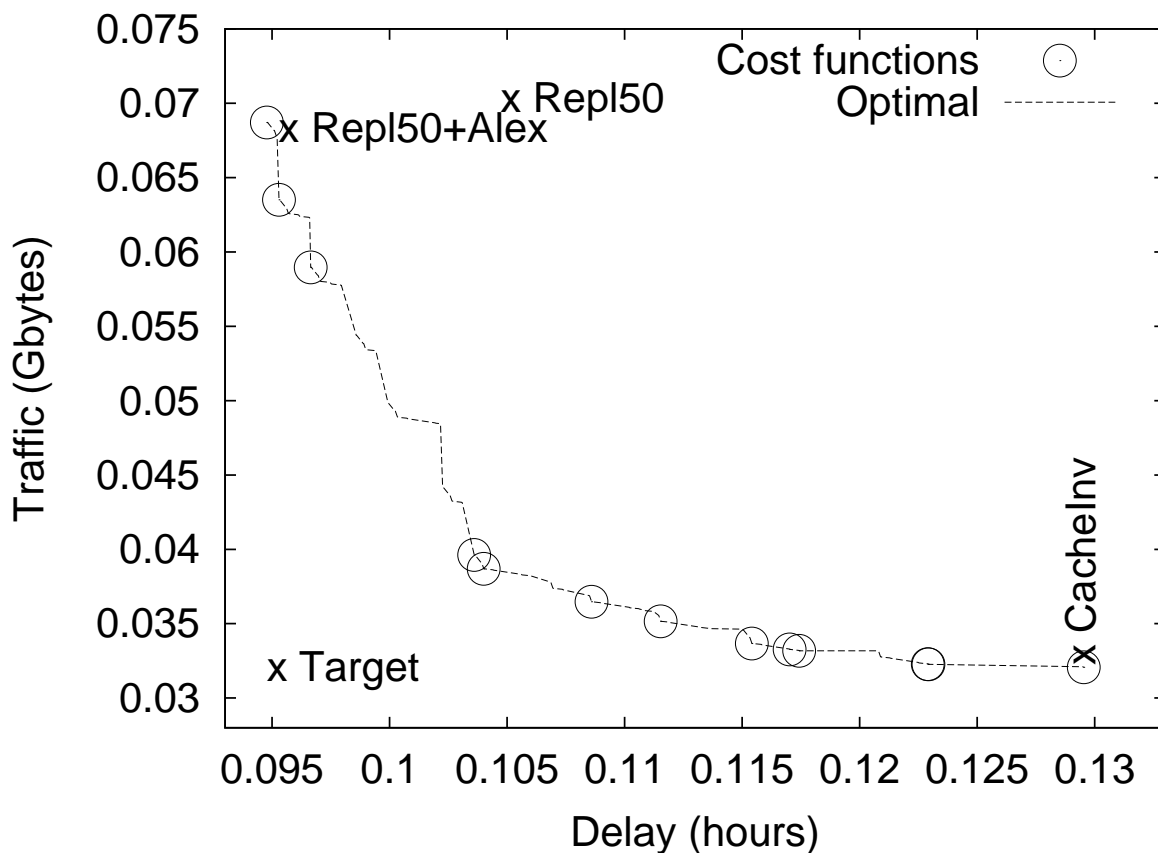
	Delay (sec.)	Incons. (#docs)	Bandwidth (Mbytes)
No repl	788553	5	44960
CacheInv	337144	5	23736
Repl50+Alex	243122	971	48058
Optimal	247387	5	23944

Strategy	Nb of docs
No repl	1286
Check	0
Alex	614
AlexCheck	0
CacheInv	530
Repl10	3652
Repl25	2345
Repl50	5420
Repl50+Alex	3279
Repl50+AlexCheck	242
Total	17368

Improving Assignment Method

Question: How well is our method for assigning strategies to documents: have we found the best possible arrangements using our cost function?

Answer: We iterated over *all* possible arrangements for a total of eight documents and selected the best one. We also calculated cost function values:



Adaptive Web Documents

Problem: Now that we know that differentiating strategies makes sense, we only need to figure out how to apply it to the real world.

Model: We make the following assumptions concerning Web documents:

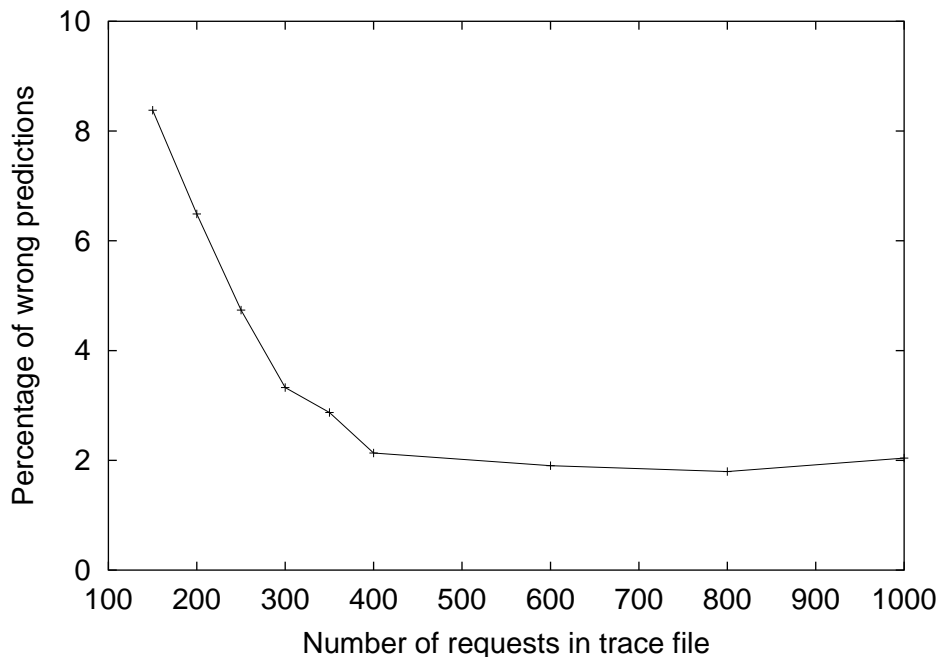
- Each document is a collection of Web pages, including icons, buttons, images, etc.
- Each document has an owner who is responsible for initiating updates.
- There is a master location from where updates take place. Only the owner updates a document.
- There are several copies located at different places that can only be read by clients.

Basic approach: We collect traces and do runtime analysis using our cost-function model for assigning strategies.

Adaptive Web Documents (2/2)

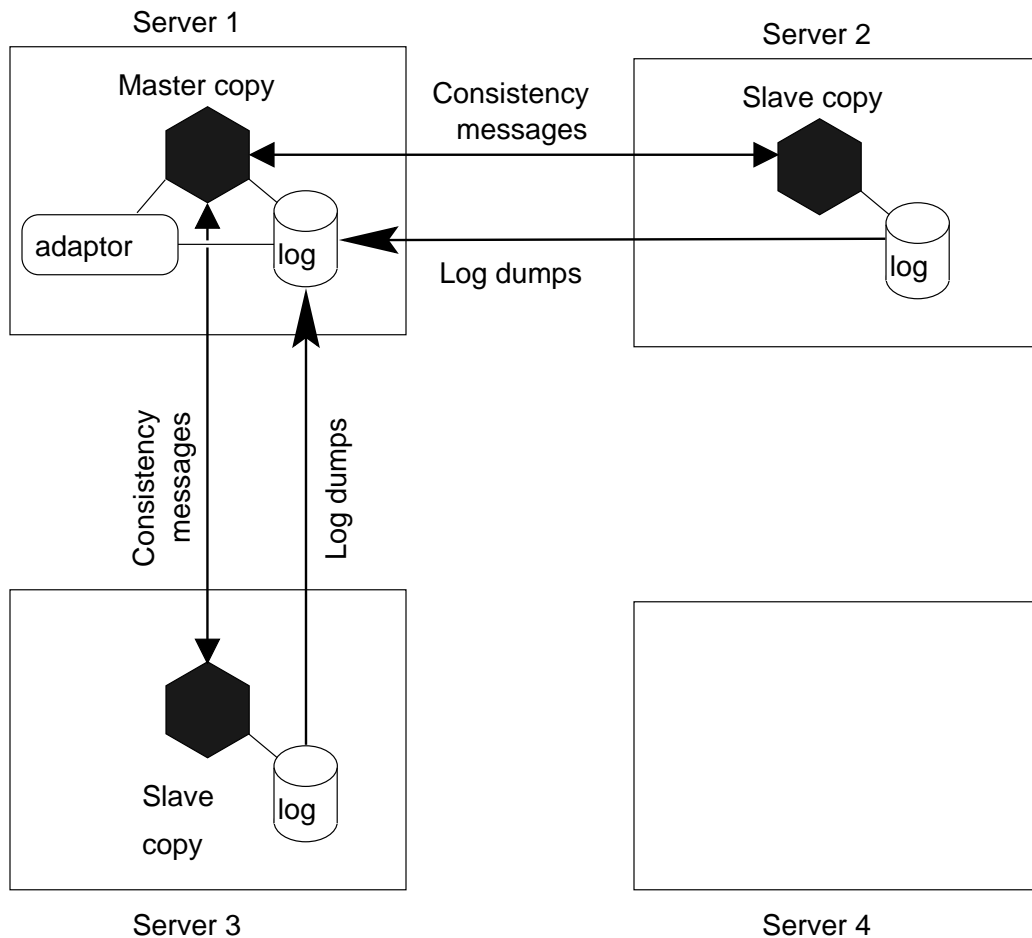
Question: how large should the traces be to give “accurate” results?

Evaluation: We chopped trace files into consecutive chunks, each containing N requests, and assigned a strategy based on the requests from a single chunk k . If that strategy was the same as the one found for chunk $k + 1$, the assignment for chunk k was considered correct.



Conclusion: A chunk of size 500 requests will do.

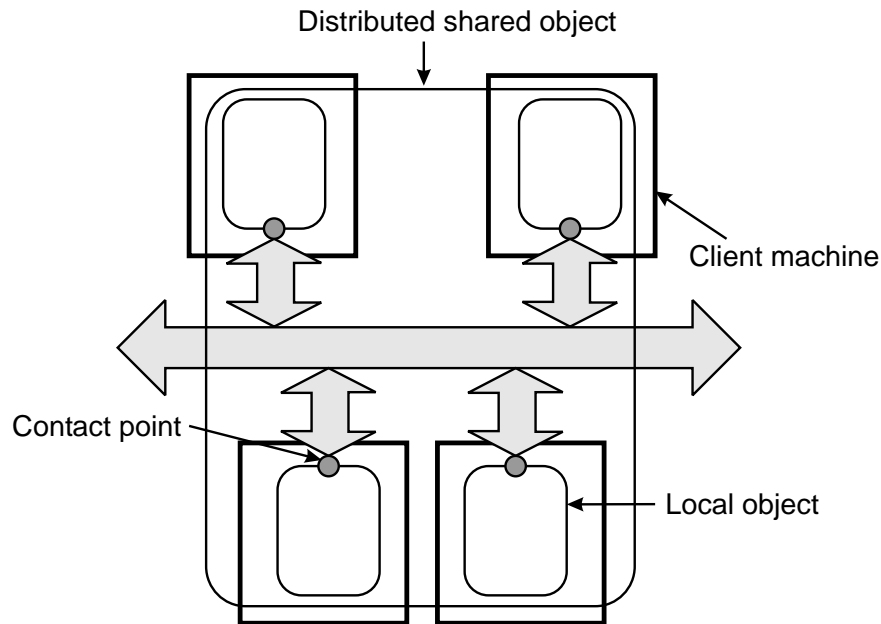
Global Architecture



All copies and the master log requests to the document. The master has a special component that decides on whether or not the current strategy should be adapted.

Encapsulating a Strategy (1/3)

Essence: Each document is constructed as a **Globe physically distributed shared object**: the object's state may be physically distributed across several machines

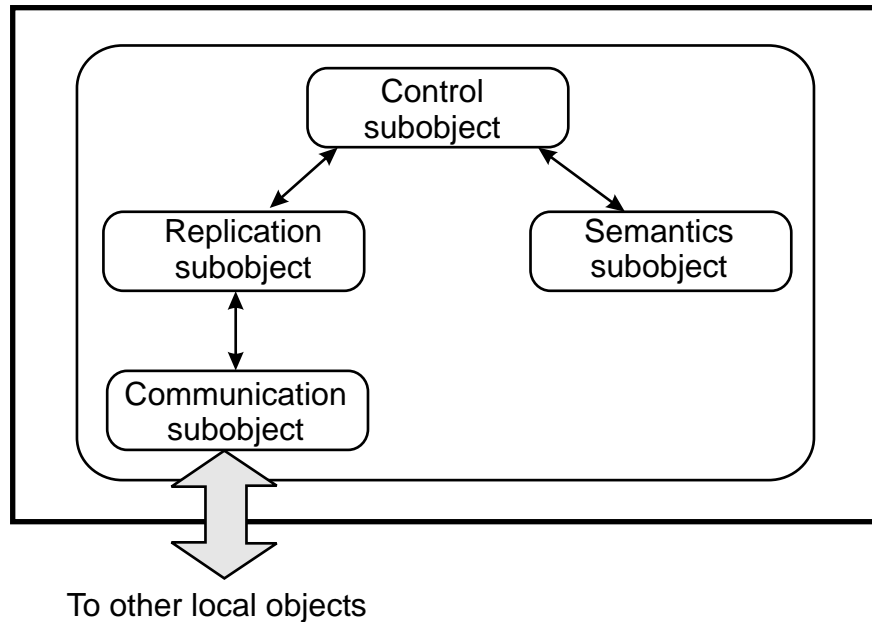


Local object: A nondistributed object residing a single address space, often representing a distributed shared object.

Contact point: A point where clients can contact the distributed object; each contact point is described through a **contact address**.

Encapsulating a Strategy (2/3)

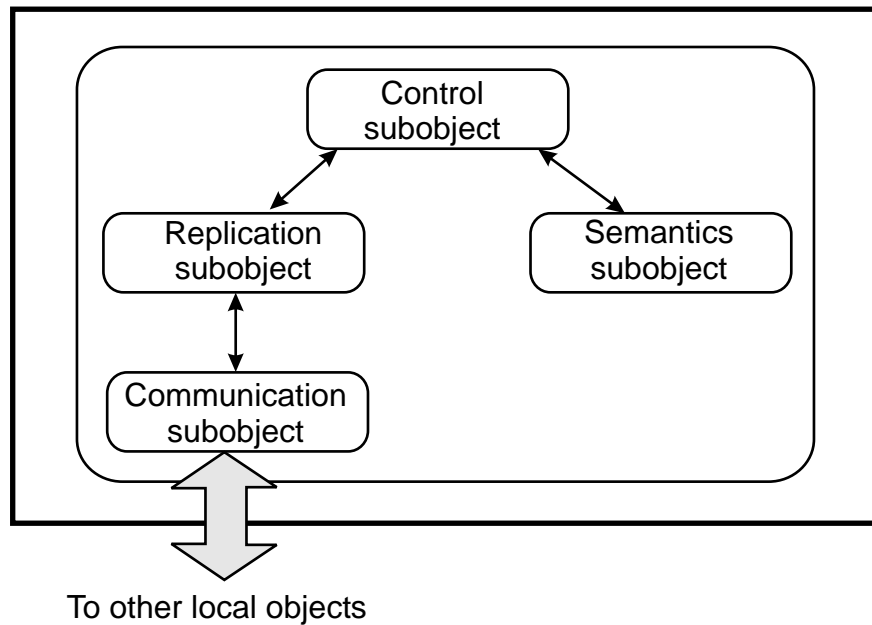
Observation: Globe attempts to separate functionality from distribution by distinguishing different local subobjects:



Semantics subobject: Contains the methods that implement the functionality of the distributed shared object.

Communication subobject: Provides a simple network-independent interface for communication between local objects.

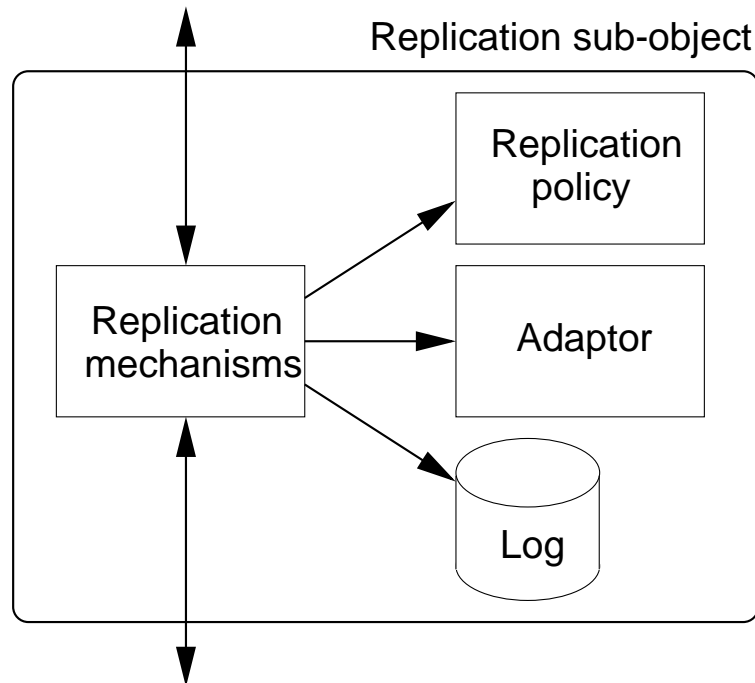
Encapsulating a Strategy (3/3)



Replication subobject: Contains the implementation of an **object-specific** consistency protocol that controls exactly when a method on the semantics subobject may be invoked

Control subobject: Connects the user-defined interfaces of the semantics subobject to the generic, predefined interfaces of the replication subobject

Replication Subobject (Master)



Log: Contains the local traces from all copies

Policy: Contains parameter values that dictate a specific replication strategy

Adaptor: Computes the best strategy

Mechanisms: Contains the actual replication algorithm (possibly dynamically downloaded).

Feasibility

Question: How feasible is this approach? Using our current trace files:

- Computing 10 different strategies for a single document based on a trace file with 500 requests takes about 140ms on a 600 MHz Pentium-III machine.
- Network bandwidth hardly increases: only a fixed overhead of approximately 50 bytes per request for logging purposes. The number of messages should, however, be reduced.

Conclusions

- It makes sense to differentiate distribution strategies for Web documents, and possibly other Internet applications as well.
- We need mechanisms to associate a distribution strategy with a single document, and to be able to make on-the-fly adaptations.
- There is a need for accurate data and/or models on the usage of various Web applications.
- Using on-the-fly trace-driven simulations appear to be workable.