Internet-based Collaborative Decentralized Systems

Replicating for performance

Maarten van Steen

vrije Universiteit amsterdam

2 of 47

Introduction

Observation: We can distinguish three **scaling dimensions** when talking about distributed systems:

Numerical: many users, many resources

Geographical: components comprising the system are placed far apart

Administrative: the system is simultaneously managed by many different administrative organizations

Problem areas: numerical scalability can often be dealt with by throwing in more hardware. Geographical and notably administrative scalability pose problems.

3 of 47

Scaling Techniques (1/2)

Distribution: Spread the work that a distributed system needs to do across multiple machines. Distribution is good for **numerical scalability**:

- The Domain Name System (DNS): nicely distributed name space
- The World Wide Web: “perfectly” distributed collection of Web pages

Scaling Techniques (2/2)

Replication/Caching: Place copies of your data close to where they are needed, so that the client-perceived performance is satisfactory. Replication is good for **geographical scalability**:

- Caching Web proxy servers: reduce client access times
- Content Delivery Networks (CDNs): provide high availability and good access times for large data files

4 of 47

Scaling Technique Problems (1/2)

Distribution: Take into account that data may have been placed really far apart.

Distribution may introduce a geographical scalability problem

Replication: When having multiple copies around, updates need to be "immediately" propagated to maintain consistency.

Caching and replication may introduce a global synchronization problem

Note: Things just get worse with **concurrent updates**.

5 of 47

Scaling Technique Problems (2/2)

Solution: We need to take into account how distributed and replicated data are used:

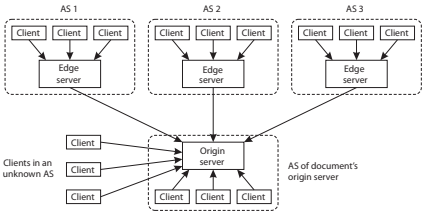
- Placement of data should be such that we can still **exploit locality**
- Replication should be done in a way that frequent global synchronization can be avoided (i.e., try to go for **weak consistency**)

Observation: We are talking about **application-specific solutions** for building scalable systems

6 of 47

An Experiment (1/3)

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?



7 of 47

An Experiment (2/3)

- We collected traces on requests and updates for all Web pages from two different servers (in Amsterdam and Erlangen)
- 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.

8 of 47

An Experiment (3/3)

Issue	Site 1	Site 2
Start date	13/9/1999	20/3/2000
End date	18/12/1999	11/9/2000
Duration (days)	96	175
Number of documents	33,266	22,637
Number of requests	4,858,369	1,599,777
Number of updates	11,612	3338
Number of ASes	2567	1480

9 of 47

Distinguished Strategies: Caching

Abbr.	Name	Description
NR	No replication	No replication or caching takes place. All clients forward their requests directly to the origin server.
CV	Verification	Edge servers cache documents. At each subsequent request, the origin server is contacted for revalidation.
CLV	Limited validity	Edge servers cache documents. A cached document has an associated expiration time before it becomes invalid and is removed from the cache.
CDV	Delayed verification	Edge servers cache documents. A cached document has an associated expiration time after which the origin server is contacted for revalidation.

10 of 47

Distinguished Strategies: Replication

Abbr.	Name	Description
SI	Server invalidation	Edge servers cache documents, but the origin server invalidates cached copies when the document is updated.
SUx	Server updates	The origin server maintains copies at the x most relevant edge servers; x = 10, 25 or 50
SU50 + CLV	Hybrid SU50 & CLV	The origin server maintains copies at the 50 most relevant edge servers; the other intermediate servers follow the CLV strategy.
SU50 + CDV	Hybrid SU50 & CDV	The origin server maintains copies at the 50 most relevant edge servers; the other edge servers follow the CDV strategy.

11 of 47

Trace Results: One Global Strategy

Turnaround time and bandwidth in relative measures; stale documents as fraction of total requested documents.

	Site 1			Site 2		
Strategy	**Turnaround**	**Stale docs**	**Bandwidth**	**Turnaround**	**Stale docs**	**Bandwidth**
NR	203	**0**	118	183	**0**	115
CV	227	**0**	113	190	**0**	**100**
CLV	182	0.0061	113	142	0.0060	**100**
CDV	182	0.0059	113	142	0.0057	**100**
SI	182	**0**	113	141	**0**	**100**
SU10	128	**0**	**100**	160	**0**	114
SU25	114	**0**	123	132	**0**	119
SU50	102	**0**	165	114	**0**	132
SU50+CLV	**100**	0.0011	165	**100**	0.0019	125
SU50+CDV	**100**	0.0011	165	**100**	0.0017	125

***Conclusion**: No single global strategy is best*

12 of 47

Per-Document Assignment (1/3)

Approach: Assume we have k performance metrics $m_1, \ldots, m_k$.

- Let D be a set of documents, S a set of strategies
- Let $\mathbf{w}$ be a weight vector (i.e., $\sum w_i = 1, w_i \geq 0$).
- Let $res(m_i, d, s)$ be the value in metric m_i for document d under strategy s

Arrangement: A set of *(document, strategy)*–pairs: $A = \{(d, s_A(d)) | d \in D, s_A(d) \in S\}$. Each arrangement has an associated **cost**:

$$cost_{\mathbf{w}}(A) = \sum_{i=1}^{k} w_i \cdot \Big(\sum_{d \in D} res(m_i, d, s_A(d)) \Big)$$

13 of 47

Per-Document Assignment (2/3)

Note: With a global strategy, $s_A(d)$ is the same for every document d.

Goal: If A is an arrangement, we are looking for an assignment of strategy $s_A(d)$ for each document $d \in D$, such that we minimize $cost_{\mathbf{w}}(A)$.

Brute-force approach: Simply try each strategy for every document and find out which combination works best. This approach essentially requires $|S|^{|D|}$ evaluations.

14 of 47

Per-Document Assignment (3/3)

Assignment rule: We reach our goal if we assign the strategy s that minimizes expression $E1$ instead of $E2$:

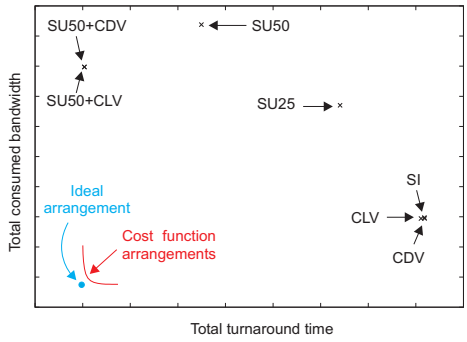
$$\underbrace{\sum_{d \in D} \sum_{i=1}^{k} w_i \cdot res(m_i, d, s)}_{E1} = \underbrace{\sum_{i=1}^{k} w_i \sum_{d \in D} res(m_i, d, s)}_{E2}$$

For the family of costs under all possible weight vectors, we get **cost function arrangements**.

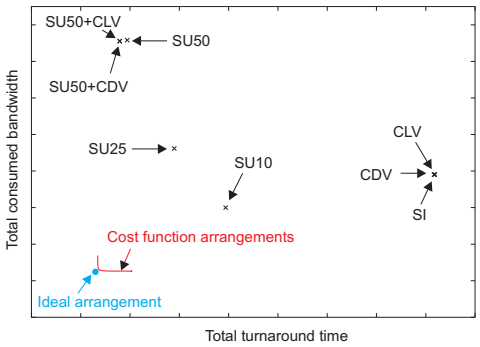
Note: This is good news, because we need to evaluate only at most $|D| \cdot |S|$ strategies, instead of the $|S|^{|D|}$ required by a brute-force approach $\Rightarrow$ we can even conduct evaluations at **runtime**!

15 of 47

Results: Site 1



Results: Site 2



Useful Strategies

Fraction of documents to which a strategy is assigned.

Strategy	Site 1	Site 2
NR	0.0973	0.0597
CV	0.0001	0.0000
CLV	0.0131	0.0029
CDV	0.0000	0.0000
SI	0.0089	0.0061
SU10	0.1321	0.6087
SU25	0.1615	0.1433
SU50	0.4620	0.1490
SU50+CLV	0.1232	0.0301
SU50+CDV	0.0017	0.0002

***Conclusion**: It makes sense to differentiate strategies*

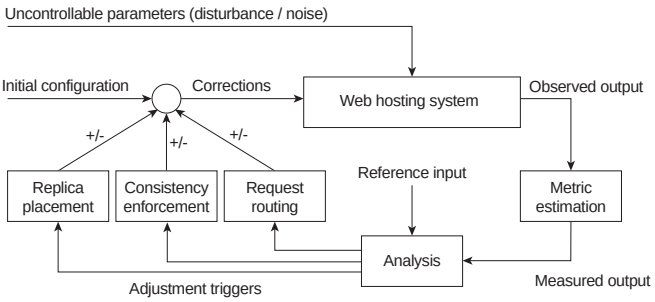
Replication for performance: requirements

- Runtime measurements in order to evaluate access patterns
- Runtime analysis to find best strategy
- Facilities to dynamically adjust:
  - location of replicated content
  - decide on number of replicas
  - change consistency enforcement
  - decide on data granularity
  - ...

We need to realize feedback control loops

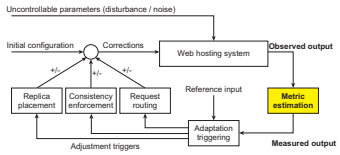
19 of 47

Replication for CDNs



20 of 47

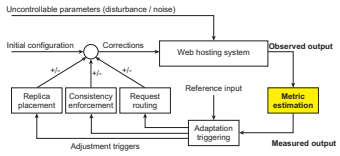
Metric determination



- Metric identification (what determines the cost of performance)
  - end-to-end latency (in time or hops)
  - total consumed bandwidth
  - consistency (value,staleness)

21 of 47

Metric determination



- Client clustering (needed for scalable measuring)
  - by proxy (e.g., DNS name servers)
  - by administrative grouping (autonomous systems)
  - network-aware clustering (essence: group by network address)
- Metric estimation (what can be measured?)

22 of 47

Metric estimation: latencies

Problem: In order to decide to which replica server requests should be redirected, we need to know how close a client is to a server:

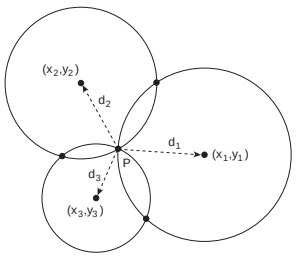
- Pinging the client is not an option: we must assume that clients do not participate.
- Clients should not be modified for latency measurements.

Solution: Try to place each node (or cluster of nodes) in an N-dimensional space, such that $d(P,Q)$ is a reasonable estimation of the latency between nodes P and Q, respectively.

23 of 47

Computing position

Observation: a node P needs $d+1$ **landmarks** to compute its own position in a d-dimensional space.



Solution: P needs to solve three equations in two unknowns (x_P,y_P):

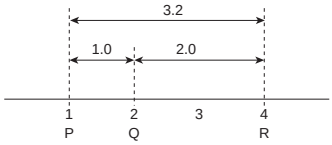
$$d_i = \sqrt{(x_i - x_P)^2 + (y_i - y_P)^2}$$

24 of 47

Computing position

Some problems:

- measured latencies to landmarks fluctuate
- computed distances will not even be consistent



25 of 47

Computing position

Solution: Let L landmarks measure their pairwise latencies $d(b_i, b_j)$, and let a coordinator compute positions by minimizing

$$\sum_{i=1}^{L} \sum_{j=i+1}^{L} \left[\frac{d(b_i,b_j) - \hat{d}(b_i,b_j)}{d(b_i,b_j)}\right]^2$$

with $\hat{d}(b_i, b_j)$ being distance to landmark b_i given a computed coordinate for b_j.

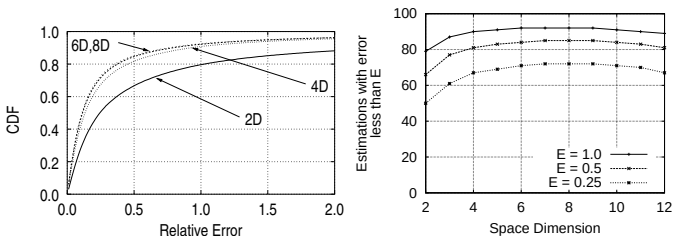
Then: let each node P minimize

$$\varepsilon = \sum_{i=1}^{L} \left[\frac{d(b_i,P) - \hat{d}(b_i,P)}{d(b_i,P)}\right]^2$$

26 of 47

Dimensions

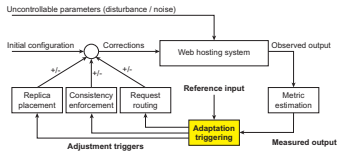
Issue: How many dimensions do we need to consider? Accept errors $\varepsilon < E$:



M = 6 dimensions turns out to be enough

27 of 47

Adaptation triggering



- Requires a reference (to determine how far we're drifting off)
  - Use cost-driven approach and minimize costs
- **Essential:** how to perform analysis
- When to activate triggers
- Which triggers to activate

28 of 47

Adaptation triggering: analysis

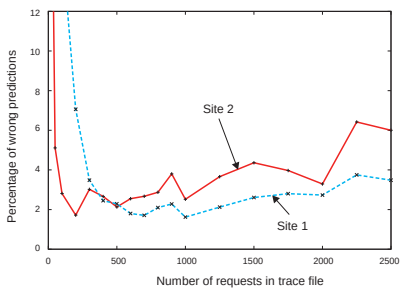
Essence: If no analytical models are available, use trace-driven simulations to different scenarios. **Example:** Globule:

- Each simulation run (i.e., strategy evaluation) for a single Web page took 140 ms
- Keeping track of strategy transitions (and thus reducing strategies for evaluation) allows for more than 10-fold improvement

Conclusion: Runtime what-if analysis is doable, assuming the number of strategies to evaluate is limited. **Note:** Measurements show that even "boring" sites require continuous strategy adaptations.

29 of 47

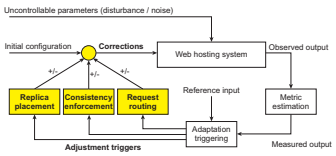
Determining trace length



Conclusion: Length of 500 requests is close to optimal. **Is it?**

30 of 47

Adaptation measurements



- Replica placement: server/content placement
- Consistency enforcement: models/policies/content distribution
- Request routing: HTTP-based versus IP-based

31 of 47

Latency-driven replica placement

Essence: Identify the region from which most requests come, and place a replica server in that region:

- Divide an M-dimensional space into cells with edge length C.
- Simply count the number of nodes (for requests) that come from which cell.
- **Problem:** Clusters of nodes may span cell boundaries.
- **Solution:** Use zones: cell plus its 3^M neighboring cells and count densities. We may now have overlapping zones.

Issue: determining the best cell size.

32 of 47

Determining cell size

Experiments with real data, combined with regression analysis leads to:

$$C = \frac{1}{8} \cdot \frac{D}{\sqrt[3]{K}}$$

with

D: Estimated average distance between nodes (requires only relatively few samples to be accurate)

K: Number of required replicas

33 of 47

Complexity analysis

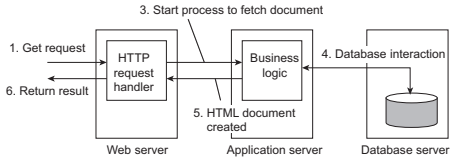
- Estimating D: $O(1)$
- Construction of zones: $O(N \log(N))$ (N = number of nodes)
  - Assign nodes to cells: $O(N)$
  - Compute zones from nonempty cells: bounded by $O(N)$
  - Access (3^M neighboring) cells by binary search: bounded by $O(\log(N))$
- Allocate K replicas by finding most dense zones: bounded by $O(K \cdot N)$

Note: Known alternatives require $O(K \cdot N^2)$ or comparable (i.e., have an $O(N^2)$ component). Computing placements with zones is consistently in the order of a few seconds compared to hours for the alternatives. **Important:** another runtime solution!

34 of 47

Web applications

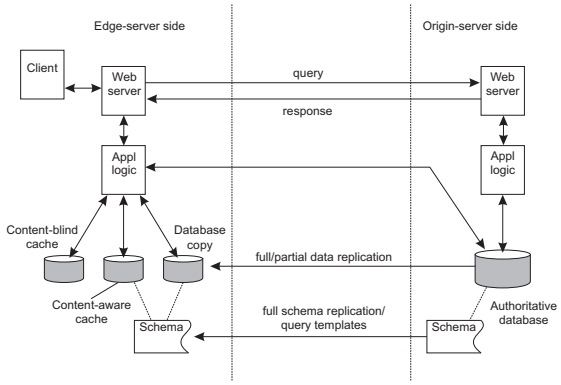
Issue: So far, we've been considering only **static content**. Real Web sites use a different architecture:



Problem: How can we replicate content that is generated on request?

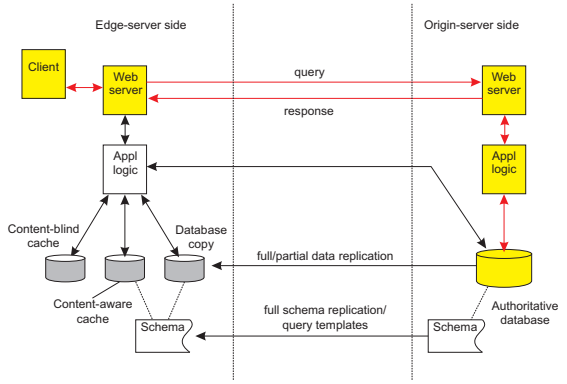
35 of 47

Edge-server systems



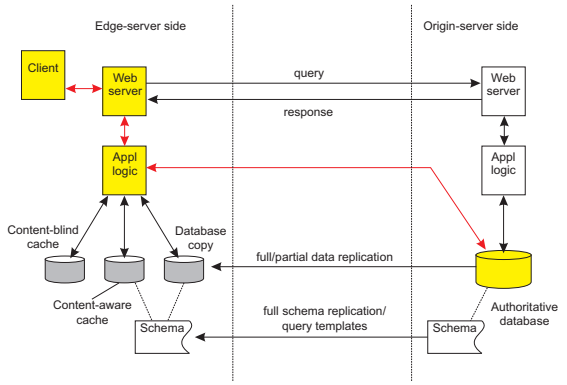
36 of 47

Standard processing



37 of 47

Application server only



38 of 47

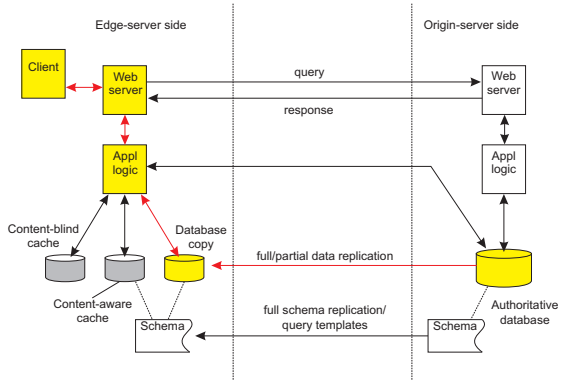
Application server only

- Local application server analyzes request
- Local application server prepares specific database query
- Direct communication with authoritative database
- **Variation:** (near) read-only data has been migrated to edge server, assuming updates are processed offline.

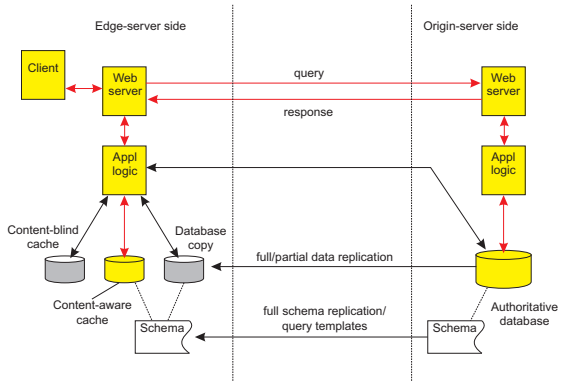
Result: offloading computations from origin server to edge servers.

39 of 47

Full/Partial DB replication



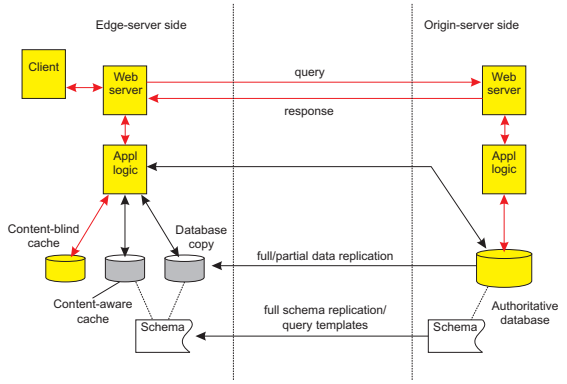
Content-aware caching



Content-aware caching

- Queries are associated with templates
- Store templates at edge servers (i.e., form local data model)
- Store query results locally, perform query-containment check
- Example:
  - Query $Q1$: Select books from author A with publication date $\leq D1$
  - $Q1$ is sent to authoritative database; result is stored locally
  - Query $Q2$: Select books from author A with publication date $\leq D2 \leq D1$
  - Containment check reveals that $Q2$ can be processed locally

Content-blind caching



43 of 47

Content-blind caching

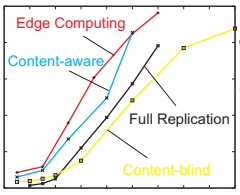
- Associate with each query Q a unique ID: *hash*(Q)
- Send Q to origin server; store (*hash*(Q), *result*(Q))-pair
- Next query Q', compute *hash*(Q') and do cache lookup

Big question: Is differentiating these strategies effective?

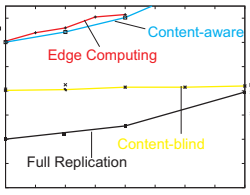
44 of 47

Workload comparisons

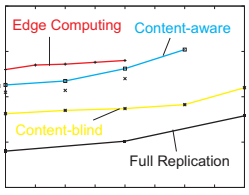
Measurements: Average client latency versus number of active clients:



Flashcrowd



Web browsing



Web ordering

45 of 47

Conclusions

- Replicating Internet-based services for performance requires differentiating replication strategies
- It is difficult to predict which strategy will be best ⇒ build feedback control loops
  - Monitor and measure systems behavior
  - Be able to analyze against ideal performance
  - Have appropriate adjustment measures
- **Note:** we need continuous feedback control

46 of 47

Reading material

[1] G. Pierre, M. van Steen, and A. Tanenbaum. “Dynamically Selecting Optimal Distribution Strategies for Web Documents.” *IEEE Trans. Comp.*, 51(6):637–651, June 2002.

[2] S. Sivasubramanian, G. Pierre, M. van Steen, and G. Alonso. “Analysis of Caching and Replication Strategies for Web Applications.” *IEEE Internet Comput.*, 11(1):60–66, Jan. 2007.

[3] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. van Steen. “Replication for Web Hosting Systems.” *ACM Comput. Surv.*, 36(3):1–44, Sept. 2004.

47 of 47