Saperlipopette!: a Distributed Web Caching Systems Evaluation Tool

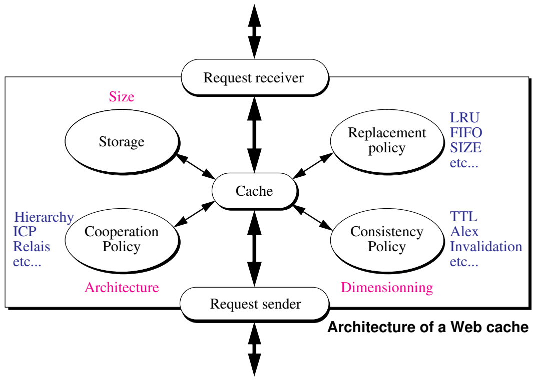
Guillaume Pierre

INRIA SOR group
http://www-sor.inria.fr/projects/relais/

8 January 1999

Introduction

- **When designing a distributed Web caching system, you have to make a number of choices.**



Optimization criteria

- **Several conflicting optimization criteria**
  - ▹ Documents' access latency
  - ▹ Documents' consistency
  - ▹ Network load
  - ▹ Money
  - ⇒ You may want to choose your optimization tradeoffs
- **Question: how to make the best configuration choice?**

Configuring a caching system

- **Using standard configurations**
  - ⊕ Generally "not too bad" results
  - ⊖ Doesn't take into account:
    - Actual Web traffic characteristics
    - Actual network connectivity
  - ⊖ Doesn't optimize your prefered metrics
- **Using custom configurations**
  - ⊕ Resulting performance can be optimized
    - Regarding your local traffic and connectivity
    - Regarding your prefered metrics
  - ⊖ Configurations are hard to determine
  - ⊖ The "optimal" configuration can change with time

Saperlipopette!

- **Saperlipopette! evaluates the performance of distributed Web caching architectures**
- **It works in two phases**
  1. Collect information about the particular target environment
     - About the Web traffic
     - About the network QoS
  2. Simulate a number of candidate configurations
- ☞ **For each configuration, it outputs the resulting performance**

Uses of Saperlippette

- **Saperlipopette allows one to choose a cache configuration**
  1. Capture the traffic of the target organization
  2. Simulate a "reasonable" configuration
  3. Refine the configuration
  4. etc...
- **Saperlipopette helps designing new caching policies**
  1. Imagine a new caching policy (e.g. a cooperation policy)
  2. Implement it in the simulator
  3. Simulate it
  4. Identify its strengths/weaknesses
  5. Refine your policy
  6. etc...

Outline of the talk

Our metrics

- **Access performance**
  - Hit ratio / Neighbor hit ratio
  - Latency / retrieve time / subjective bandwidth
    - Absolute / relative values
    - Distribution curves
- **Consistency**
  - Percentage of out-of-date documents delivered by the cache
- **Network resources**
  - Traffic Mb

Data collection

- **Capturing a site's WWW traffic**
  - From the logs of a proxy
  - Separate collection of additional data
- **Evaluating the wide-area network QoS**
  - Via simple measurements

WWW traffic collection [1/2]

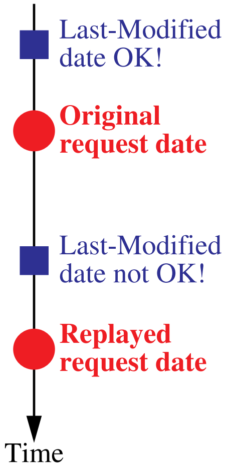
- **The simulator needs the following input data:**
  1. Client address
  2. Requested URL
  3. Date of the request
  4. *Ideal* timestamp of the document
     → Timestamp that a *perfectly consistent* cache would deliver
  5. *Ideal* size of the document
  6. Network QoS bewteen the cache and the server

☞ **Only 1, 2 and 3 are in the proxy logs**

☞ **We must collect 4, 5 and 6 separately**

WWW traffic collection [2/2]

- **To retrieve the ideal timestamp and size:**
  - Every night, a robot replays the requests of the day
  - It sends `HEAD` requests directly to the servers
  - If the document was not updated since the "real" download, we get the desired info
  - We treat the other situation with "best" and "worst" case strategies



Network QoS collection

- **We measure separately the network QoS**
  - By issuing regular GET requests to the servers
  - By measuring:
    - The HTTP latency
    - The stream bandwidth
- **This a rather naive approach**
  - We believe it is accurate enough for first approximation
  - It is easy to replace the network simulation modules

☞ **We can use instead "standard" traces (such as DEC traces)**
  - But no info on the site's distribution

The simulator

- **The simulator takes as input:**
  - A number of trace files
  - A network performance representation
  - The Web caching configuration to evaluate
    - One server answers all missed requests
    - The caches can be configured
      - Internal policies (replacement, consistency, ...)
      - Dimensioning
      - Distributed architecture
    - Each cache has one client which executes a trace file
- **It outputs:**
  - The desired metrics
- **It is easy to integrate new policies to be simulated**

Example config file

```
dates=(841300000,843500000);

server[0]=();
cache[0]=(server[0], SimClient("netperf.txt"),
          Ttl(1), Fifo(500), Store(200),
          NoCoop);
client[0]=(cache[0], ConstClient(5,300000),
           Scenario("headers-dec.txt"));

run();

report(HIT_RATIO,cache[0]);
report(LAT_DIST,client[0]);
```


Methodology

- **Used metrics :**
  - Performance : relative retrieve time
  - Consistency : % of outdated delivered documents
- **Optimization criterion :**
  - Maximiser performance
  - But also try to keep a reasonable consistency
- **Method :**
  - Vary one parameter at a time: cache size, consistency policy
  - At each step, restart from identical configs for both traces

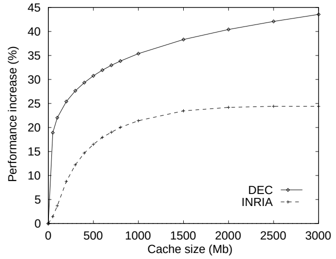
☞ **Show that these traces need different cache configuration.**

Trace files characteristics

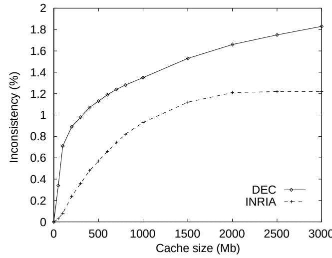
	INRIA	*DEC*
Duration	150 days	25 days
Nb of requests	1,617,916	24,658,773
Nb of unique requests	1,014,390	3,934,419
Theoretical max Hit rate	37,3%	84,0%
Nb of client machines	1,089	17,354
Avg size of documents	12.1 ko	11.5 ko
Avg retrieve time without cache	9.3 s	3.6 s

Influence of cache size

- **Replacement : LRU — Consistency : Alex(0.3)**



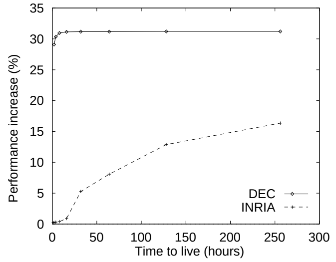
(a) Performance increase



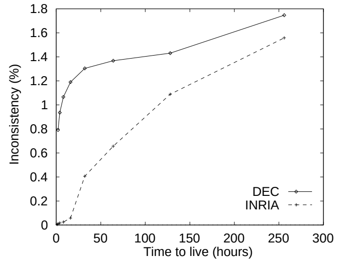
(b) Consistency

Influence of the consistency policy [1/3]

- **Cache size : 500 Mb — Replacement : LRU**
- **TTL: fixed time to live**



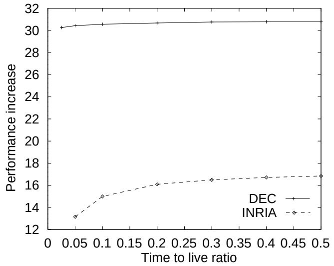
(c) Performance increase



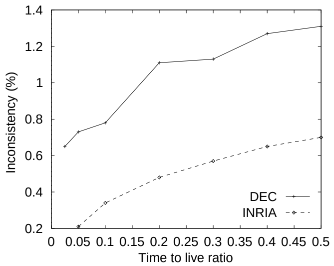
(d) Inconsistency

Influence of the consistency policy [2/3]

- **Alex: Time to live proportional to the document's age**



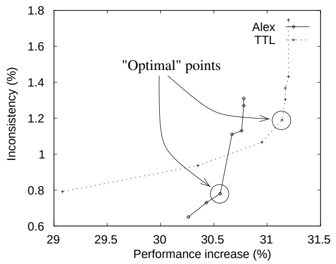
(e) Performance increase



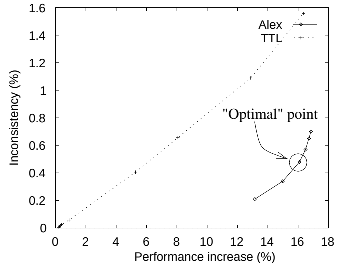
(f) Inconsistency

Influence of the consistency policy [3/3]

- **Reachable performance/consistency compromises**



(g) DEC



(h) INRIA

Conclusion

- **Saperlipopette! allows one to:**
  1. Monitor WWW activity
  2. Evaluate the performance of candidate caching configurations
- **Saperlipopette is free software. It is available at** `http://www-sor.inria.fr/projects/relais/`
- **It has already been in use**
  - In “École Nationale Supérieure des Télécommunications de Bretagne” for simulating Web caches over networks with resource reservation
  - By “Compagnie des Signaux” for a prospective simulation of the French Internet over the next 4 years.