Orestes: a REST protocol for horizontally scalable cloud database access

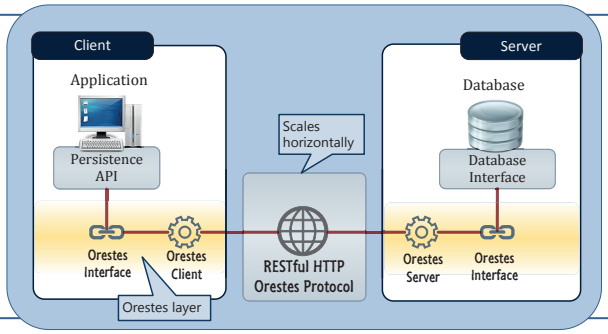
Felix Gessert, Florian Bücklers, Norbert Ritter
University of Hamburg

Observations

Open problems:
Database as a service, cloud computing
⇒ *network latency*
Web-scale applications, commodity hardware
⇒ *scalability issues*

Goal:
Scalable, cloud capable database systems

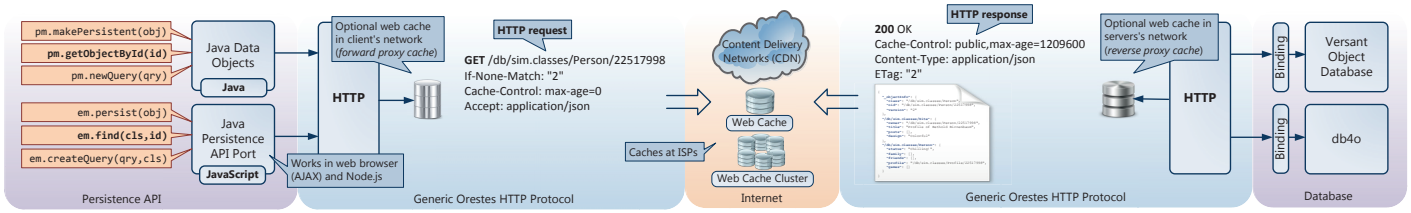
Our approach:
ORESTES (*O*bjects *REST*fully *e*ncapsulated in *s*tandard formats)



Solution

Property	Mechanism
✔ **Read scalability**	Web caching of database objects
✔ **Low latency**	Cache deployment in client network
✔ **Loose coupling of persistence API and DB**	Generic HTTP access protocol and resource structure
✔ **Standard formats**	Extensible HTTP content negotiation + default JSON formats

System Overview



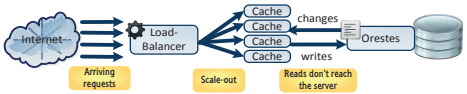
Caching mechanism:
HTTP caching with statically assigned object lifetimes

Synchronization:
Optimistic concurrency control to deal with stale objects

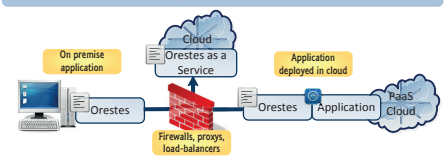
Implementation:
Generic network layer, 2 persistence APIs, 2 DB bindings

Illustration of the Key Properties

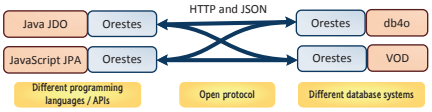
Reads scale horizontally



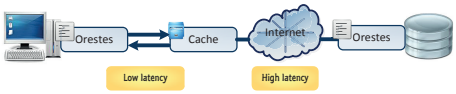
Cloud and service capable DB interface



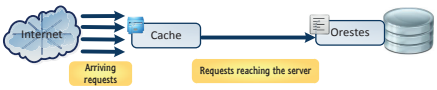
Arbitrary combinations of persistence API and DB



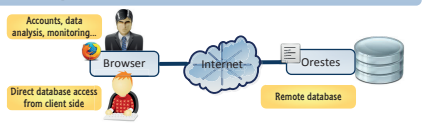
Low latency



Reduction of server load



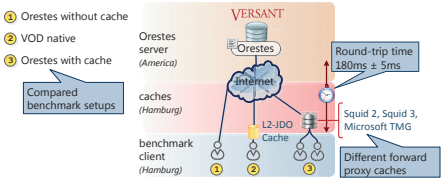
Browser persistence and administration



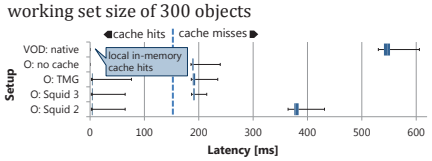
Benchmarking the Protocol

Scenario	social networking
OO model	inheritance, aggregation, etc.
Access pattern	transactional navigation
Database	Versant Object Database (VOD)
Persistence API	Java Data Objects (JDO)
Protocols	VOD TCP, Orestes
Caches	JDO L2 (VOD), web caches (Orestes)
Concurrency	single client, 50 parallel clients

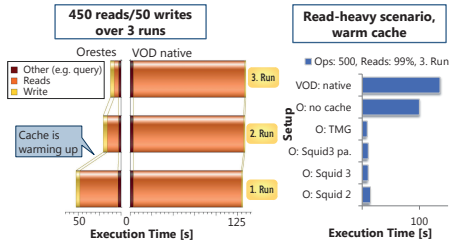
Single Client Scenario



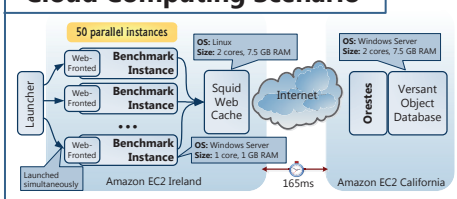
Benchmark procedure: starting with cold cache, performing 50/500 operations in 3 consecutive runs, using a read/write ratio of 50/90/99% and a working set size of 300 objects



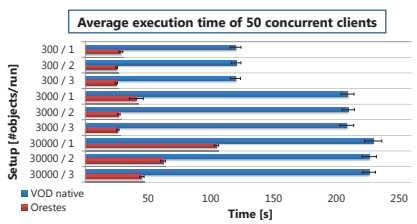
Effects of cache hits: reducing the latency of object access by orders of magnitude



Cloud Computing Scenario



Benchmark procedure: starting with cold cache, 500 operations, 3 consecutive runs, read/write ratio of 90%, 300/3000/30000 total database objects, 50 concurrent clients



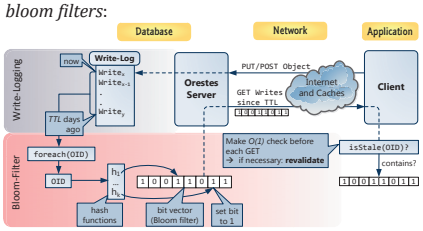
Conclusions

We achieved:
- Read scalability through web caching
- Considerable reduction of network latency
- Increase of read performance through caching of working sets

Orestes: *a novel approach to achieve read scalability for transactional object oriented persistence*
Open problem: *write scalability*

Future Work

Preventing stale reads and transaction aborts using *bloom filters*:



Further caching support: server-side invalidation for content delivery networks and server-controlled reverse proxy caches.

Further Information

◀ Visit us at the URL *http://orestes.info* or contact us via email: *research@orestes.info*