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 (Objects RESTfully encapsulated in standard formats)

**Solution**

- **Property**
  - Read scalability
  - Low latency
  - Loose coupling of persistence API and DB
  - Standard formats

- **Mechanism**
  - Web caching of database objects
  - Cache deployment in client network
  - Generic HTTP access protocol and resource structure
  - 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**
- **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

**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**
  - 100 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

**Future Work**

- Preventing stale reads and transaction aborts using bloom filters

**Further Information**

Visit us at the URL [http://orestes.info](http://orestes.info) or contact us via email: research@orestes.info