JBoss Certified Technology Partner

Israel JBoss User Group

Session 01 / 16.3.2006

Case Study, Hibernate 2nd Level Cache

By : Shachar Segev “Pontis”

The Pontis Integrated Marketing System

A system for *definition*, *immediate execution* and *management*
of targeted ***Packages***, ***Bundles*** and ***Promotional Offerings***
based on ***real-time*** individual customer behavior

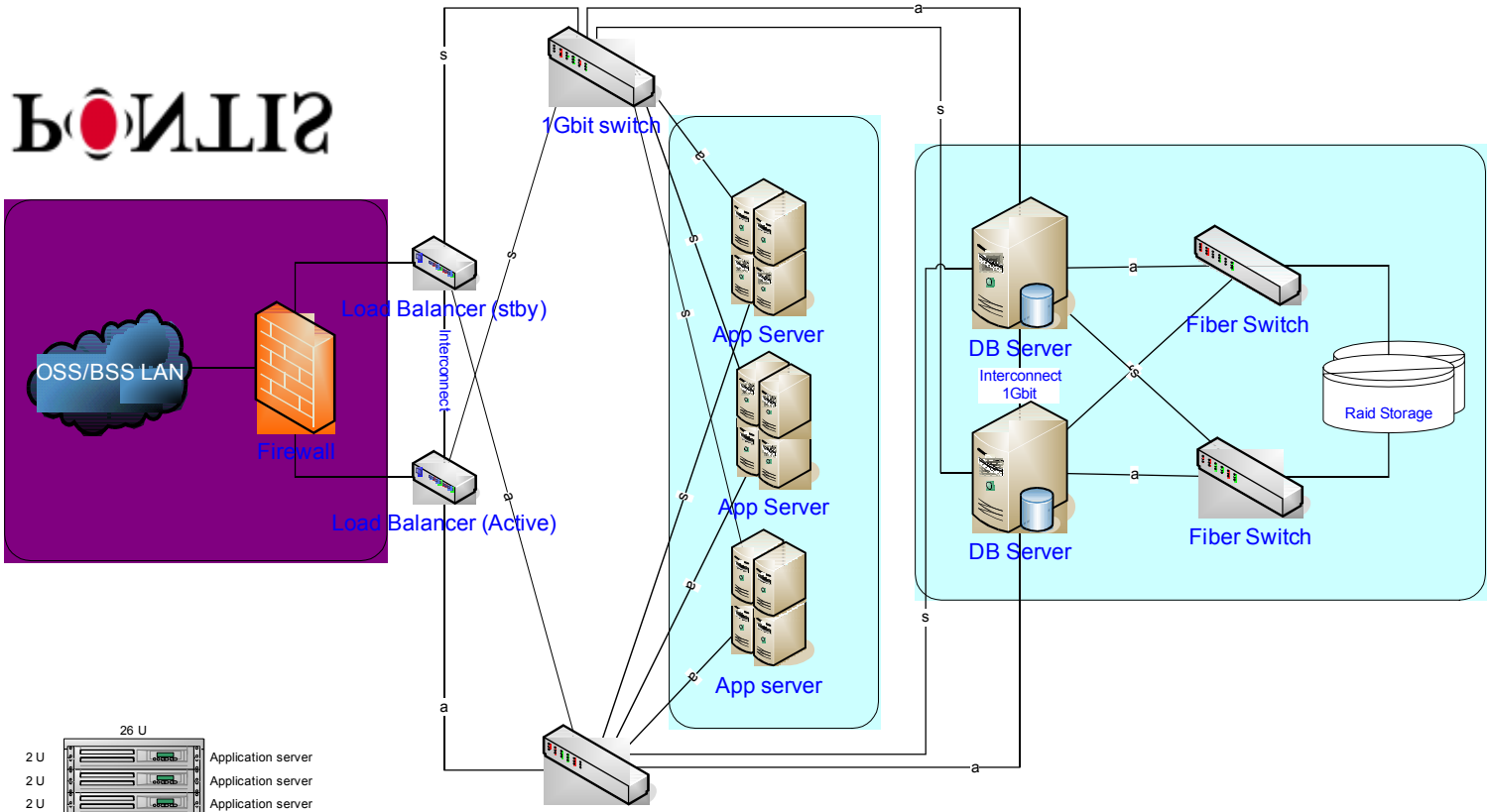
to achieve

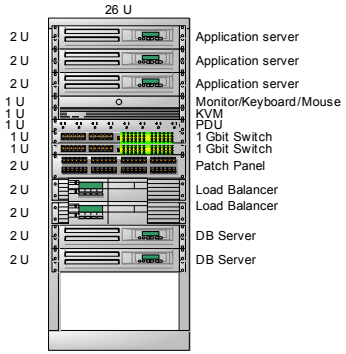
Life-time value increase,
accelerated service adoption
and ***enhanced bonding*** with customers

Performance requirements

- Hundreds requests per second
- 40 requests per second for single cpu
- Average response time 100 m/sec
- Response time under 200 m/sec
- Each request:
  - Access thousands of records
  - Update 3-8 records

Physical architecture





WSD AS1
8FE , 128MB

HP ProCurve Switch 2724
(or Cisco 2950 like)

J2EE Application Server
Tomcat Web Server
Windows 2003 Server
DL380 Like H/W
Single XEON CPU (up to 2)
2GB RAM (up to 8-12GB)
Dual Fan
Dual Power Supply
2 SCSI Hard Drives @ RAID 1
Dual Port NIC

ORACLE 9i Enterprise Edition
Real Application Cluster
Technology
Linux/Windows 2003 Server
DL380 Like H/W
Single XEON CPU (up to 2)
4GB RAM (up to 8-12GB)
Dual Fan
Dual Power Supply
2 SCSI Hard Drives @ RAID 1
Dual Dual Port NIC
Dual Fiber Card

500 GB Storage

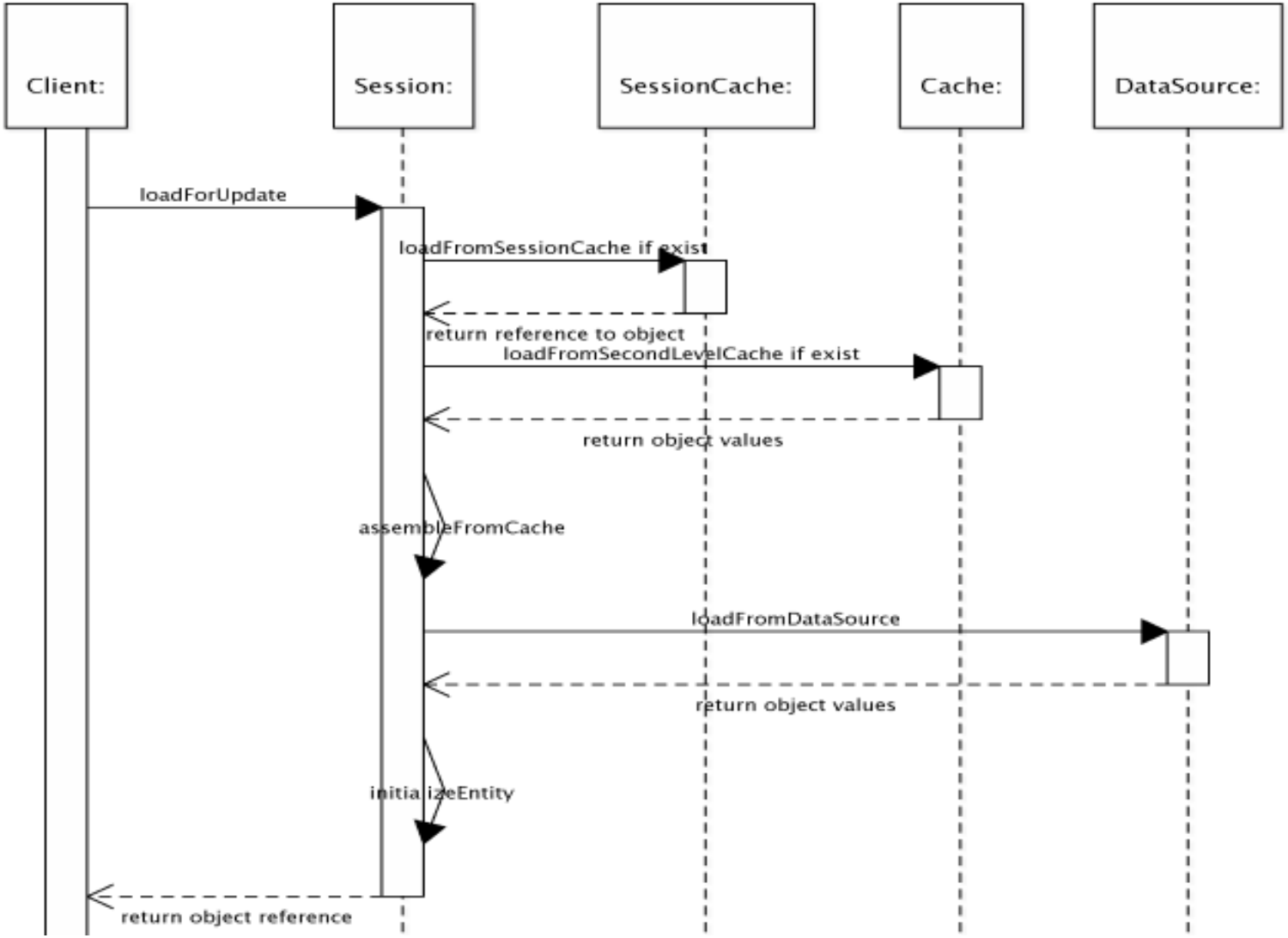
3rd party products & tools

- Java/J2EE
- Open source libraries/CM
  - Eclipse, Ant, Junit, CVS
  - Spring
  - Quartz (scheduler)
  - Log4j
  - EHcache
- Hibernate for DB persistency
- Struts + .Net controls for web UI
- Dundas for charts
- Axis for Web services
- Portability
  - OS – Windows, Linux, Solaris
  - App server – Jboss, Web logic, Web sphere
  - DB – Oracle, SQL server, DB2

The Problem

- In hibernate each thread that query data from the DB get it's own copy (instance) from the first level cache
- Creating huge amount of objects in each request cause very long response time (few seconds in our case)
- UI lists that query coarse grain objects have similar problem

Load For Update



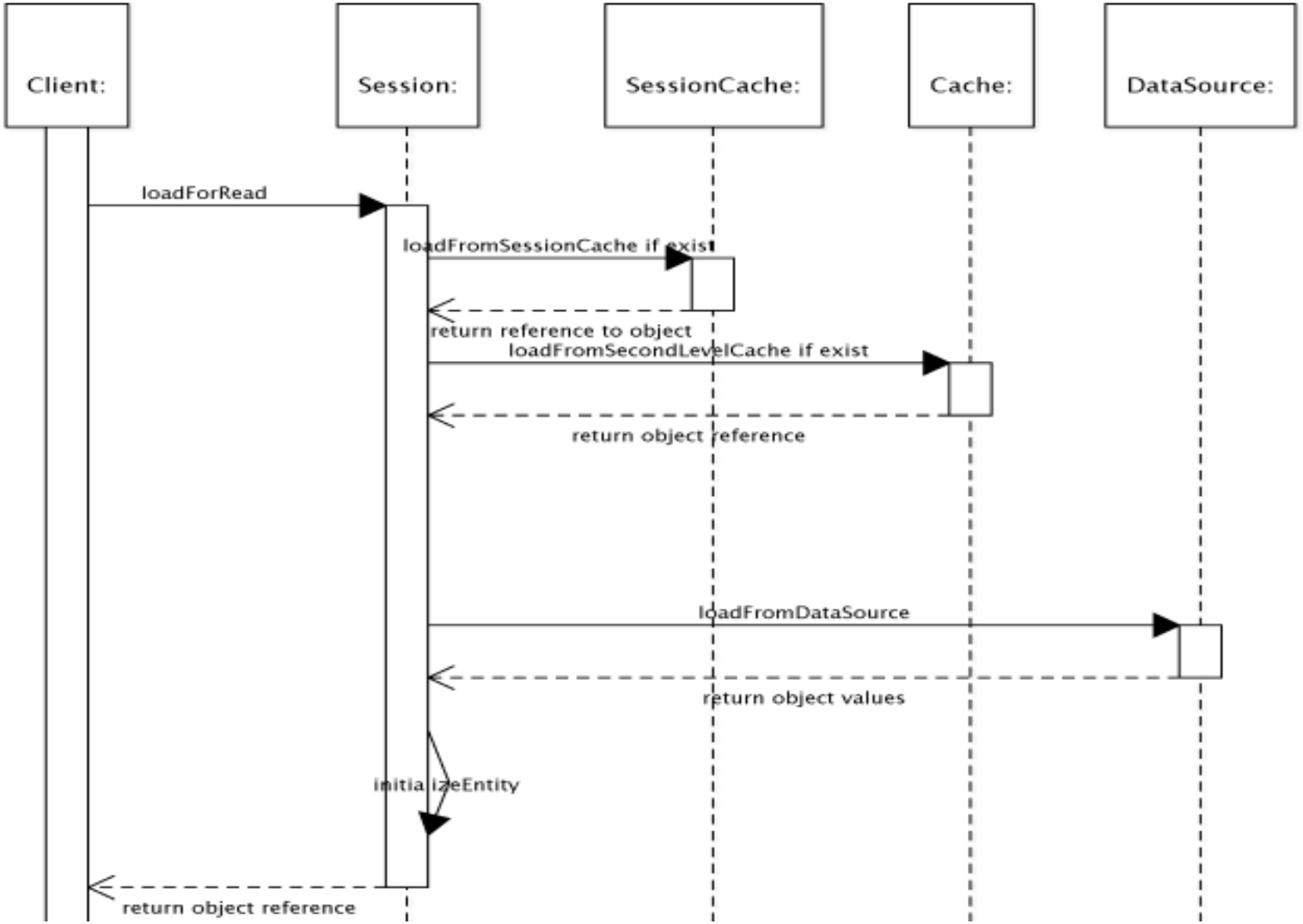
Sharable entities

- Queries return read only entities
- Read only entities throw exception when called to setters
- Read only entities are kept in the central store (sharable cache) and referenced by all threads that Query them
- Entity.loadForUpdate return updatable copy
- References from updatable copy are to read only Entities
- Updates cause next queries to return updatable version of the Entity

Implementation

- Hibernate hold in the second level cache assembled entities and return reference to those entities instead of assemble new copy
- LoadForUpdate implemented as new hibernate lockMode which assemble new copy of entity from the second level cache and put it in the tread first level cache

Load For Read



Thank you

shachar@pontis.com