J2EE Performance Scalability and Clustering
Part 1

Emmanuel Cecchet
INRIA Rhône-Alpes, ObjectWeb

Goal

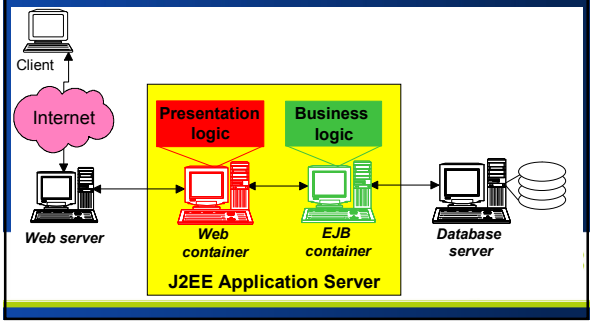
- J2EE performance scalability evaluation
- Performance factors
  - design patterns
  - container design
  - communication layers
  - Java Virtual Machine

Outline

- Background
- Performance factors
- RUBiS benchmark
- Performance results
- Scaling further with clustering
- Conclusion

J2EE architecture

- Presentation and business logic separation



Enterprise JavaBeans

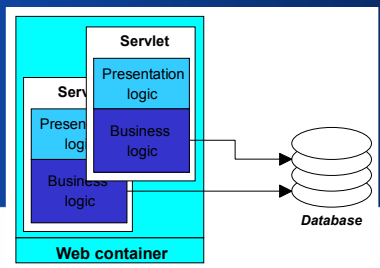
- Entity beans
  - map data stored in the database
  - Bean Managed Persistence (BMP)
  - Container Managed Persistence (CMP)
- Session beans
  - stateless: temporary operations
  - stateful: temporary objects

Outline

- Background
- Performance factors
  - design patterns
  - container design
  - communication layers
- RUBiS benchmark
- Performance results
- Scaling further with clustering
- Conclusion

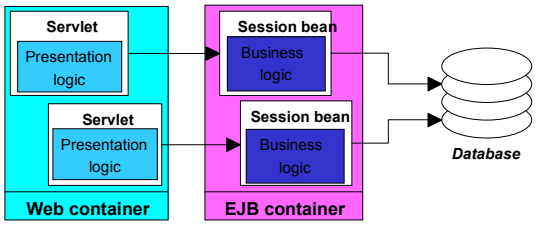
Design patterns: Servlets only

- Presentation and business logic mixed



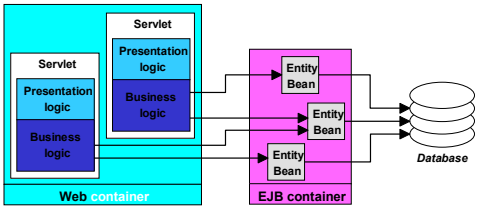
Design patterns: Session Beans

- Presentation and business logic separation



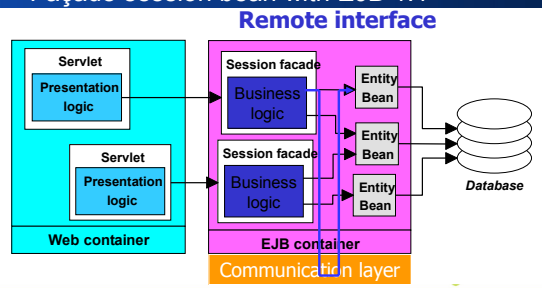
Design pattern: Entity Beans

- Data Access Objects separation with Entity Beans (BMP or CMP)



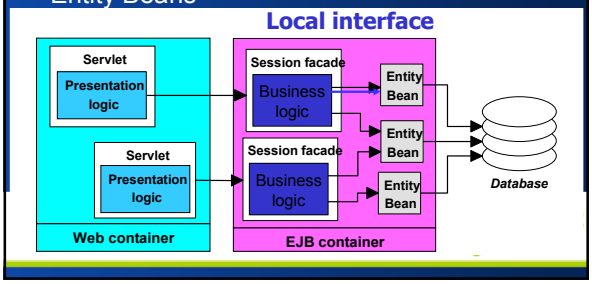
Design patterns: Session façade

- Façade session bean with EJB 1.1



Design patterns: EJB 2.0 local

- Session façade with EJB 2.0 local interface Entity Beans



Outline

- Background
- Performance factors
  - design patterns
  - container design
  - communication layers
- RUBiS benchmark
- Performance results
- Scaling further with clustering
- Conclusion

Container design

- Reflexive approach (JBoss 2.4)
  - dynamic proxy
  - home and component interfaces generated at run-time
  - reflection to locate a bean or map method signatures
- Precompiled approach (JOnAS)
  - custom implementations of home and component interfaces
  - direct call of the bean instance
  - need a specific “compiler”

Communication layers

Comm. layer	Protocol on TCP/IP	Object passing	Local calls optimized	EJB container
Standard RMI	JRMP	value	no	JBoss JOnAS
Optimized RMI	JRMP	reference	no	JBoss
Jeremie	GIOP	reference	yes	JOnAS

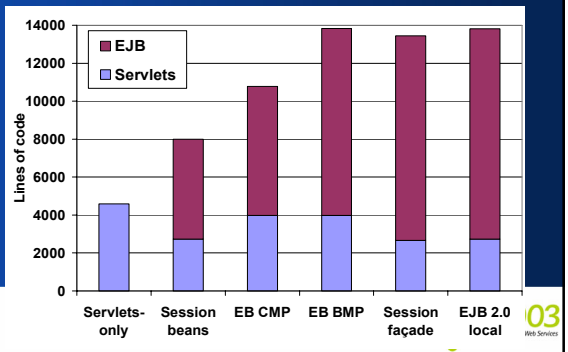
Outline

- Background
- Performance factors
- RUBiS benchmark
- Performance results
- Scaling further with clustering
- Conclusion

RUBiS

- Auction site modeled after eBay
- 26 interactions
- Browsing mix: read-only mix
- Bidding mix: 15% read-write interactions
- Database: 1.4 GB
  - 1 million users, ~500000 comments
  - >500000 items, 330000 active bids

Code complexity



Code complexity

- Beans are easy but verbose to write
- Large number of beans result in large code base
- Main portability issues
  - naming conventions
  - deployment descriptors

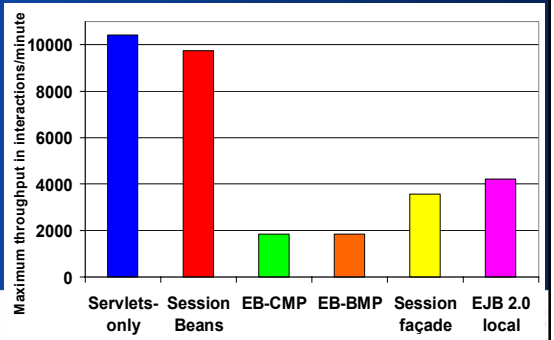
Outline

- Background
- Performance factors
- RUBiS benchmark
- Performance results
- Scaling further with clustering
- Conclusion

Design pattern evaluation

- Comparison of the 6 design patterns
- Results using:
  - Sun JVM
  - best container design
  - best communication layer
- Metric: throughput in interactions per minute

Design pattern results



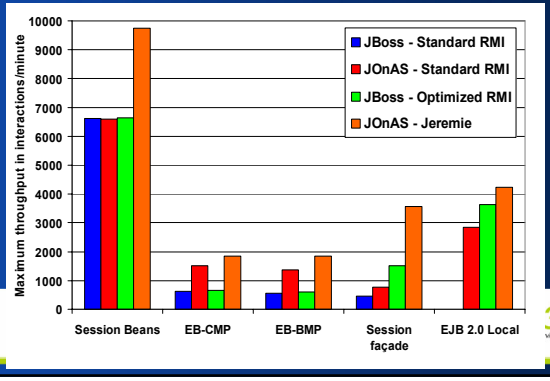
Design patterns

- Session Beans = Servlets only
- Entity Beans
  - BMP = CMP
  - data access too fine grain
- Session façade
  - allows coarser grain access
  - 2.5 times slower than Session Beans

Other performance factors

- Comparison of the 5 EJB design patterns
- Varying:
  - container design
  - communication layer
- Metric: throughput in interactions per minute

RUBiS – Overall results

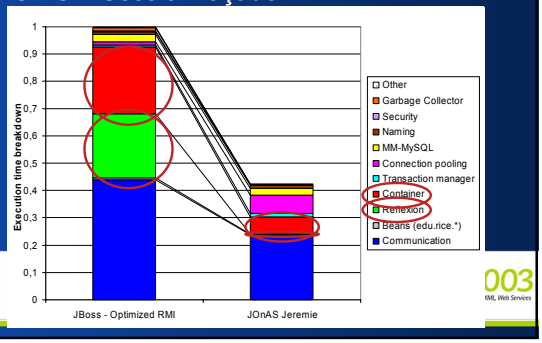


Performance factor analysis

- Container design
  - pre-compiled scales better than dynamic proxy
  - significant impact on Entity Beans

Container design

- RUBiS – Session façade

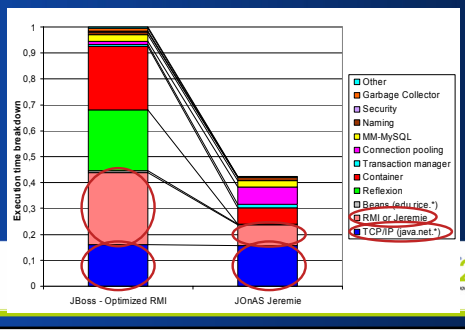


Performance factor analysis

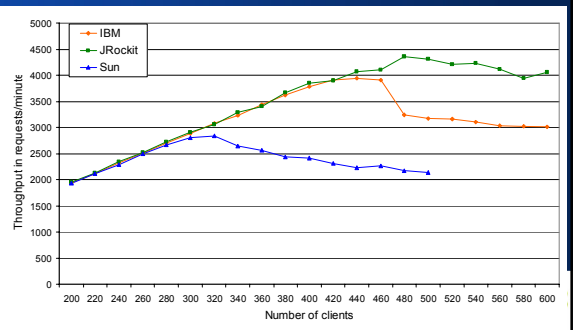
- Communication layers
  - optimized layers needed for scalability
  - EJB 2.0 local interfaces to avoid local communications

Communication layers

- RUBiS – Session façade



JVM Performance



JVM Performance

- IBM 1.3.1 is the best performance/stability tradeoff
- JRockit is very unstable and performance can dramatically drop
- Sun 1.3.1 or 1.4.x is the worst performer but is quite stable

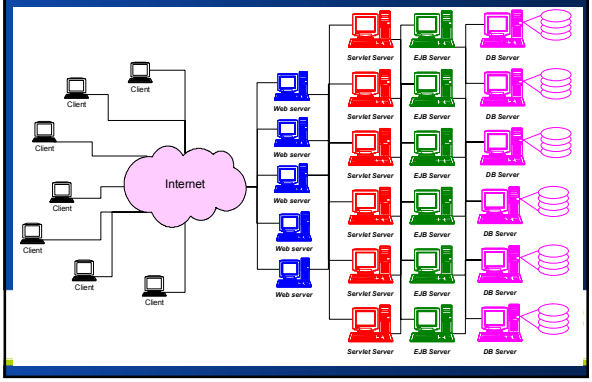
Other results

- Monitoring of CPU, disk, network and memory
- CPU profiling at the peak point using OptimizeIt
  - 45-90% spent in communication
  - up to 32% in reflection
- JDK 1.4 does not improve performance

Outline

- Background
- Performance factors
- RUBiS benchmark
- Performance results
- Scaling further with clustering
- Conclusion

J2EE Clustering



J2EE Clustering

- Web server clustering
  - hardware: L4-switch
  - software: LVS, RR-DNS, …
- Servlet/JSP server clustering
  - in-memory session replication

J2EE Clustering

- EJB Server clustering
  - cluster stubs for load-balancing
  - database based persistency
- Database clustering
  - Oracle RAC
  - Clustered-JDBC

Conclusion

- Design pattern determines performance
- Communication accounts for 45-90% of CPU time
- Reflection limits container scalability
- Less than 2% of execution time in user bean code
- JVM limits overall scalability
- Clustering can help scaling further

Some links

- RUBiS: http://www.objectweb.org/rubis
  - JMS
  - EJB 2.0 CMP
  - JDO
- JOnAS: http://www.objectweb.org/jonas
- JBoss: http://jboss.org
- JDBC: http://c-jdbc.objectweb.org

More to come on J2EE Clustering …

J2EE Performance Scalability and Clustering *Part 2*

Emmanuel Cecchet
INRIA Rhône-Alpes, ObjectWeb