Performance Scalability of EJB Applications

Emmanuel Cechet, Julie Marquerite, Willy Zwaenepoel (Rice University)

Presented by:

Vijay Lakshminarayanan

EJB

- Enterprise JavaBeans (EJB) technology is the server-side component architecture for the Java 2 Platform, Enterprise Edition (J2EE) platform. EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.
- Enterprise application developers were increasingly employing component-based software development techniques, which enable them to reduce their time to market and improve their software quality.
- Growing trend toward "componentization"

EJB

- EJB provides a number of services like DB access (JDBC), transactions (JTA), messaging (JMS), naming (JNDI), and management support (JMX).
- EJB server manages one or more ‘container’ – container responsible for providing component pooling and lifecycle management, client session management, database connection pooling, persistence, transaction management, authentication, and access control.
- 2 types of EJB – entity beans (maps data stored in the DB, one bean instance per table row), and session beans (stateless for temporary operations or stateful for temporary objects)
- Persistence maintained either in the bean – BMP (embed SQL in the bean code) or in the container – CMP (mapping between bean instance and DB column).

Design Alternatives
Session Beans

- Use session beans to implement business logic, leaving only presentation logic in servlets
- Connection pooling and transaction management by EJB server, greatly simplifying the servlets code where connection pooling may have to be manually coded.
- Ideal for short-term conversations between multiple clients and a server – store client state in stateful session beans.

Entity Beans CMP

- Extract Data Access Code from servlets and move to entity beans.
- Business logic in the servlets invoke methods on entity beans that map the data stored in the DB.
- Management of persistent identity – where many clients will look up data over a long period of time.

Entity beans BMP

- DAO separation, as before
- With BMP, SQL queries have to be hand coded in beans; while with CMP, SQL queries generated by the EJB container.
- In both cases, stateless beans to execute complex joins.
- Two different versions to track difference in persistence maintenance cost at container and at bean level.

Session Facade

- Stateless beans as a façade that abstract the entity components.
- Reduces number of business objects exposed over the network.
- Calls between façade and entity beans are local to the EJB server.

EJB 2.0 local interfaces

- EJB 2.0 optimizes intra-JVM calls to communicate between the entity beans and façade (in EJB 1.x, RMI is employed to communicate between them)
- Entity beans with a local interface cannot be called remotely, only session façade beans have a remote interface that is exposed to servlets.
- Interaction between session and entity beans therefore, bypass the communication layers in EJB 2.0.

Container

- Container provides EJB servives to a particular EJB.
- Interface between client and the bean.
- Client only interacts with home and component interfaces provided by Container, and container forwards the call to the appropriate bean.
- Beans accessed through container-generated classes.

Container design

- Pre-compiled approach – container generates custom implementations of the home and component interfaces – call appropriate method of the bean instance directly.
  More popular approach.
  Resulting classes available to client by way of the class-path or the ejb-jar file.
- Dynamic proxy based container – generate home and component interfaces at runtime by using Java Reflection.
  Map method signatures to appropriate implementations or locate bean, given the name of the class.

Implementation

- RUBiS (Rice University Bidding System) – auction site like eBay.
- 2 workload mixes – browsing mix (read-only) and bidding mix (15% read-write)

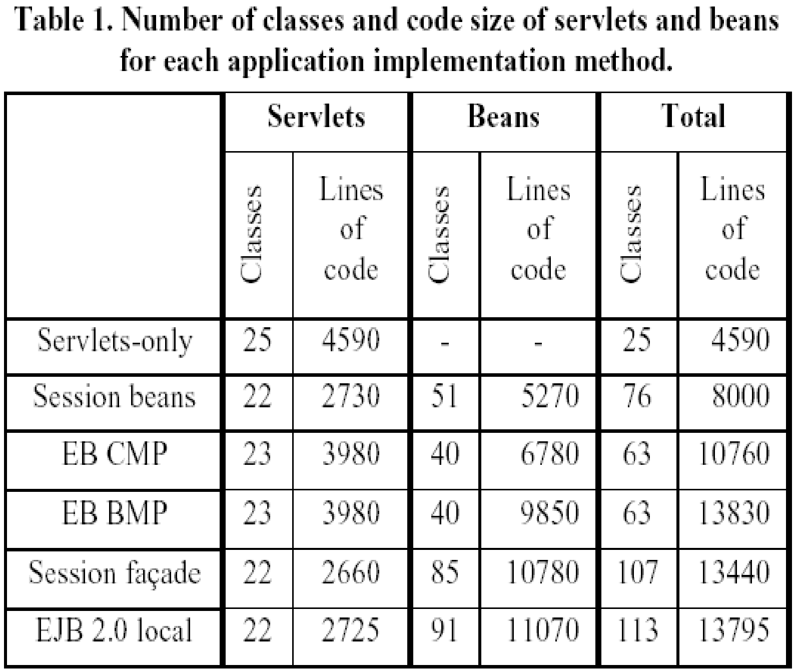

Table 1. Number of classes and code size of servlets and beans for each application implementation method.

	Servlets		Beans		Total	
	Classes	Lines of code	Classes	Lines of code	Classes	Lines of code
Servlets-only	25	4590	-	-	25	4590
Session beans	22	2730	51	5270	76	8000
EB CMP	23	3980	40	6780	63	10760
EB BMP	23	3980	40	9850	63	13830
Session façade	22	2660	85	10780	107	13440
EJB 2.0 local	22	2725	91	11070	113	13795

Implementation Key-Points

- Both EB and servlets version use servlet for user authentication. Business logic done by servlets.
- All other implementations, business logic moved from servlets to session beans and user authentication done in beans.
- Each bean requires 3 classes – home interface, remote interface, and the bean implementation.
- Entity beans provide a large number of getter-setter methods.
- Remote entity bean access not permitted in EJB 2.0 implementation.
- EJB’s are easy to write, but makes code very verbose, because the number of beans can become very large.

Experimental Results

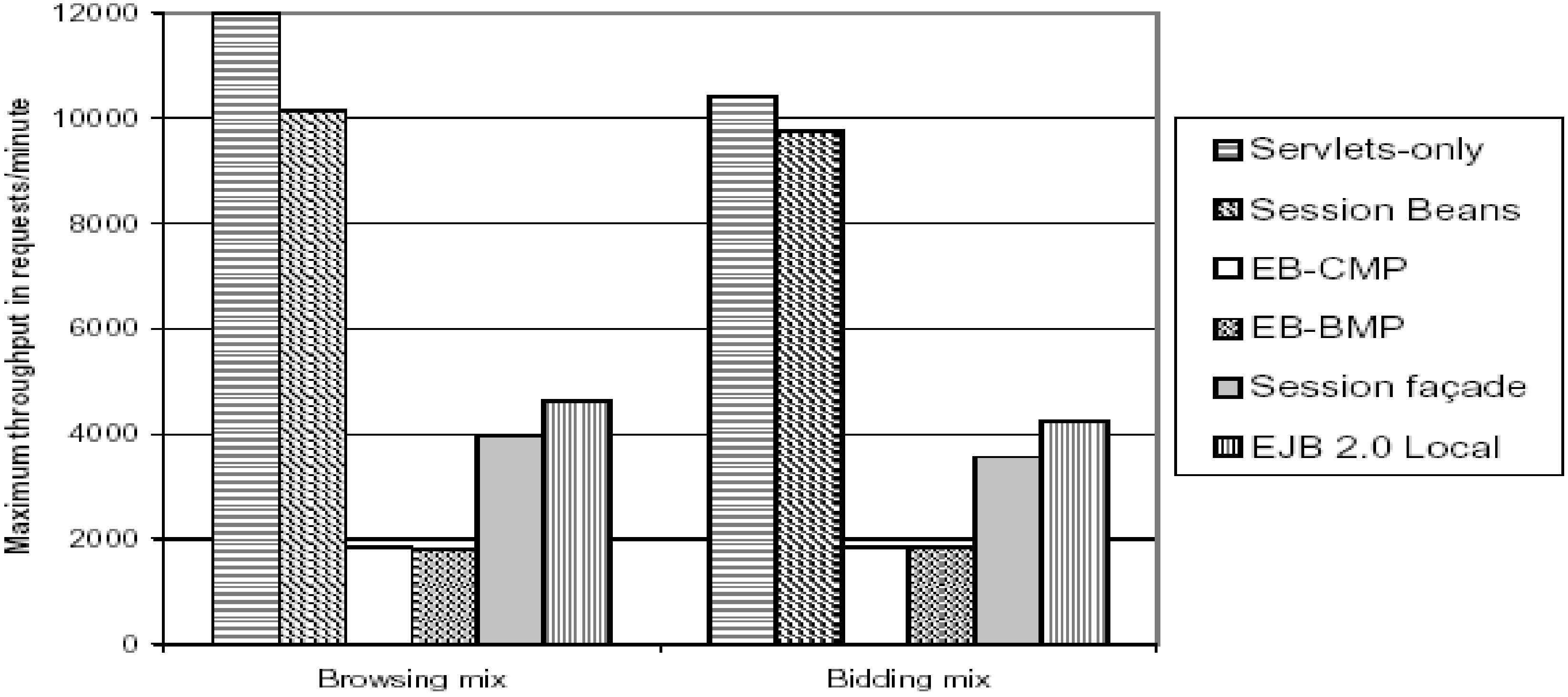

Figure 5. Maximum achievable throughput for each implementation method.

Summarized Results

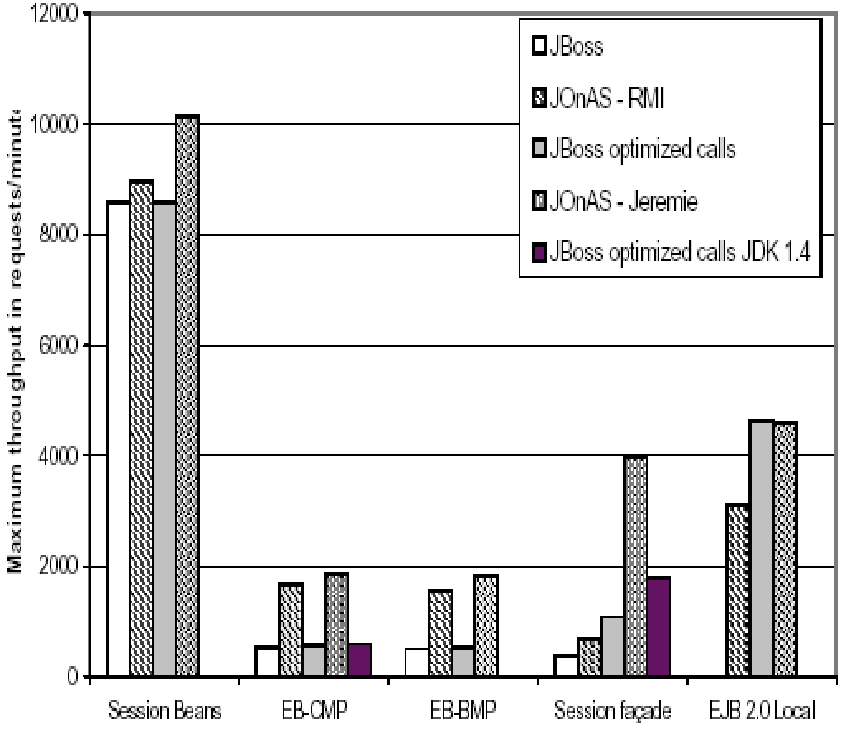

Figure 28. EJB implementations maximum throughput in interactions per minute for the browsing mix.

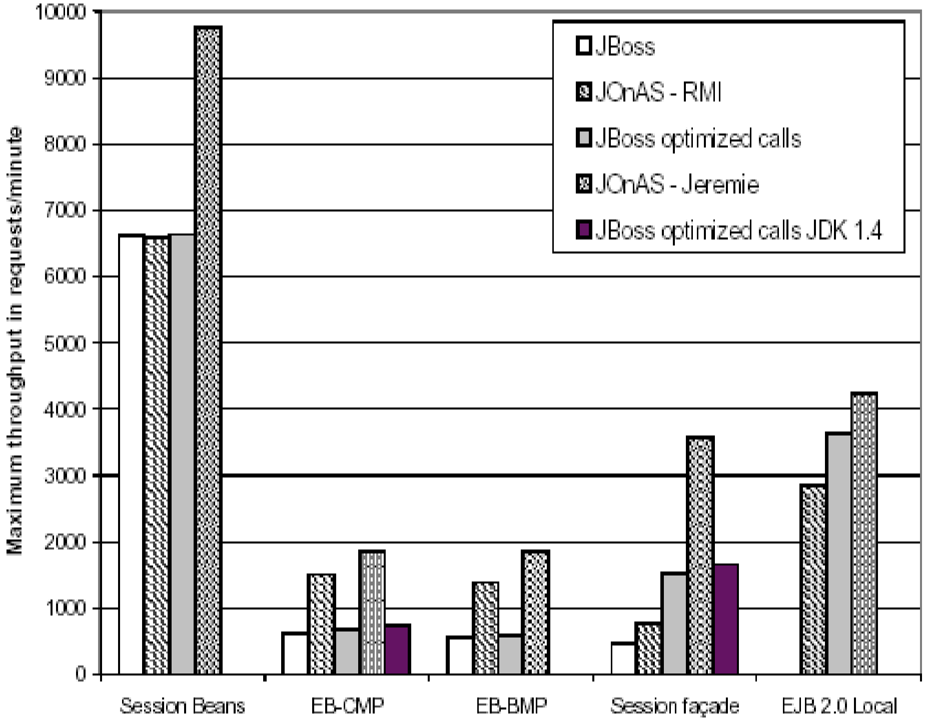

Figure 29. EJB implementations maximum throughput in interactions per minute for the bidding mix.

Key Points

- Session beans gives best throughput.
- DAO separation with EB gives least scalable resulsts. Reason: Too many remote accesses from servlets. CMP marginally better than BMP.
- Reflection overhead evident in session façade implementations.JDK 1.4 decreases Reflection costs, but increasing container cost, resulting in not much better performance.
- JOnAS-Jeremie – pre-compiled container classes with optimized communication layer.
- EJB 2.0 local interfaces performs much better than EJB 1.x session facades.
- Interestingly, bean code written by programmer represents abt 2% of execution time. Application implementation method and middleware design have greatest impact of performance.

Summary

- Stateless session beans with BMP, coupled with efficient communication layer offers performance comparable to servlets.
- EB impose row-level access to DB – lowers performance.
- Container design has no significant influence on session beans, because communication costs dominate, but has a direct impact on performance with EB.
- Dynamic proxy approach has large overhead. Therefore, pre-compiled approach – better scalability.
- Container design and local communication cost – determining factors for scalability of session façade.
- Reflection cost increases with number of beans, quickly resulting in bottleneck.
- EJB 2.0 allow RMI-based configurations to scale better and avoid communication layers for local communication.