A Java Multitenant Application Server

P. Patros[1], D. Dilli[1], K. Kent[1], M. Dawson[2], T. Watson[2]
1: Faculty of Computer Science, University of New Brunswick
2: IBM
Patros.Panos@unb.ca, Dayal.Dilli@unb.ca, Ken@unb.ca,
Michael Dawson@ca.ibm.com, tjwatson@us.ibm.com

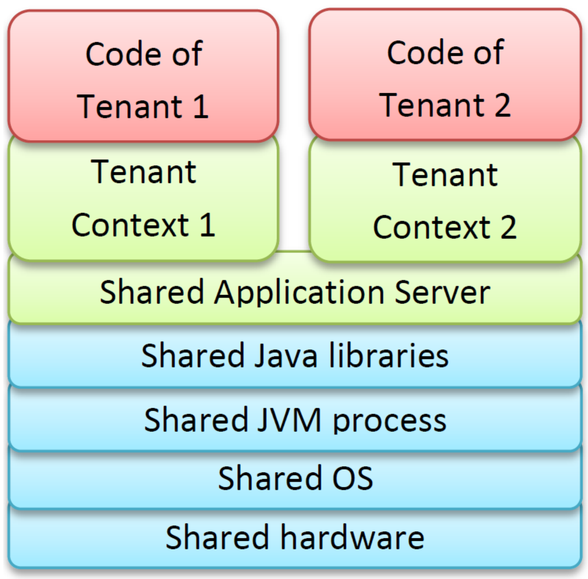
Introduction

Multitenancy enables sharing of resources between different users, also known as tenants. The tenants execute their code as if the resources were held individually by them.

We propose a technique for a multitenant application server in Java which uses a single JVM to support multiple tenants, each represented by its own JAR file without any changes to the tenants' code. We base our work on the internal multitenant features (Xmt) of the IBM JVM which we exposed with a Java API we call the Tenant API.

Our approach is significantly more efficient in saving memory (measured around 70% less) without any major throughput reductions when compared to IBM's basic multitenant mode as well as the standard one-JVM-per-tenant mode.

Furthermore, we discuss the theoretical maximum memory sharing levels across a number of different JVM configurations, including our proposed technique, using the Daytrader3 benchmark on the Liberty Web server. **The results from the two approaches converge**.

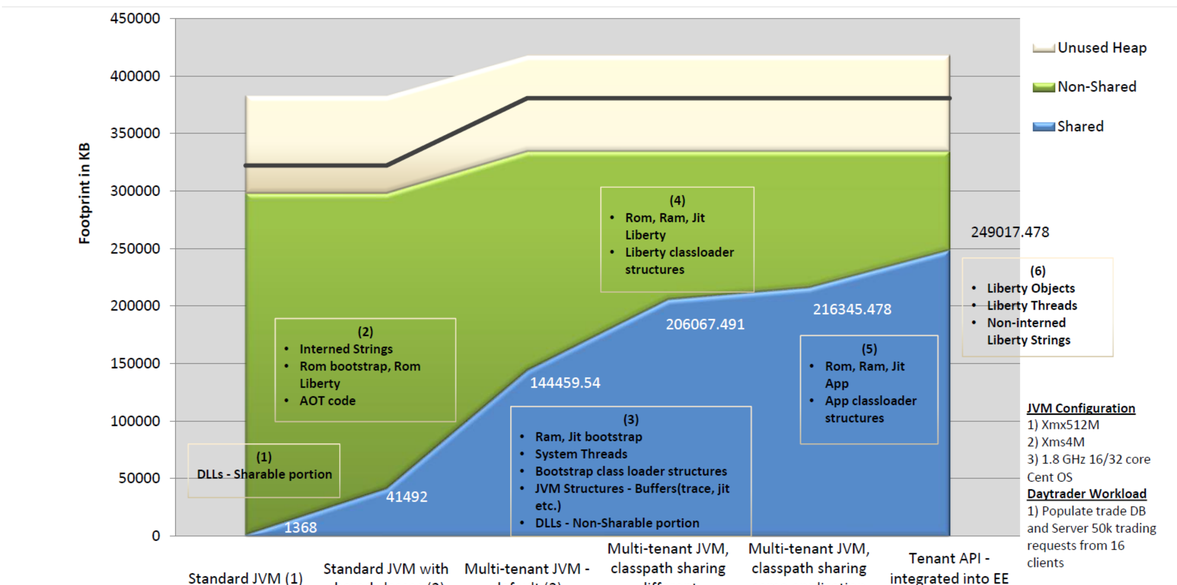


We share virtually all of the hardware and software stack between our tenants, whose code is executed on top of a private Tenant Context, ensuring static field and performance isolation.

TenantAPI

Our TenantAPI exposes the IBM JVM multitenancy which allows it to create and destroy tenants as well as run code in their context enabling performance and common statics isolation.

Theoretical Analysis of Memory Sharing

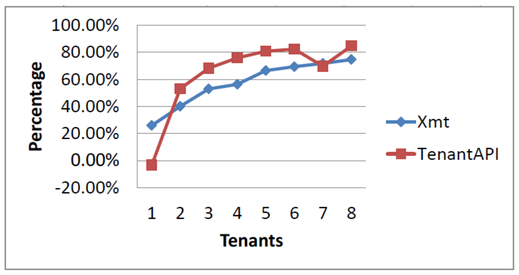


	Standard JVM (1)	Standard JVM with shared classes (2)	Multi-tenant JVM - default (3)	Multi-tenant JVM, classpath sharing different applications (4)	Multi-tenant JVM, classpath sharing same application (5)	Tenant API - integrated into EE server (6)
Unused Heap	84913	84913	83613	83613	83613	83613
Non-Shared	296882.478	256772.478	190294.938	128686.987	118409	85737
Shared	1368	41492	144459.54	206067.491	216345.478	249017.478
Resident Memory	322250.478	322264.478	380754.478	380754.478	380754.478	380754.478

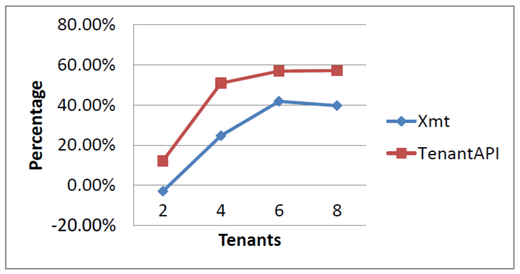
The Multitenant Server

Our server starts by scanning a folder where the tenant JARs are located, creating a new tenant context and class loader per tenant using our TenantAPI and mapping tenants to their URLs. When a request arrives, the mapping is looked up to find the tenant it belongs to and the serving code from the JAR file is run on the tenant's context using our TenantAPI.

We measured footprint savings and speedups in two applications. One hello world and another with flights and hotels bookings. We compare our version with Xmt using one JVM per tenant as the baseline.



Hello World



Flight-Hotels DB