Cloud*Tran*

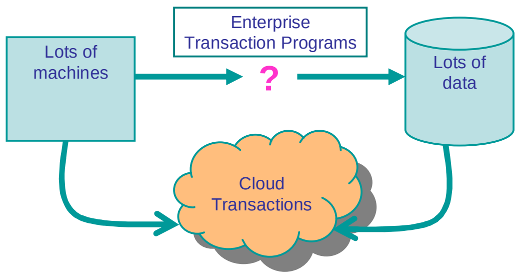
Scalable Transactions for Cloud Applications

CloudTran is a middleware product that brings the Cloud into the mainstream as an application platform for Java developers. It overcomes the inherent difficulties of distributed programming by providing a simple Java view of data, mission-critical transactions and automatic distribution in the Cloud. The combination of a cost effective infrastructure coupled with scalable and fast transactions means that CloudTran delivers higher performing applications at a reduced cost.

Distributed Application Development

Public and private clouds are increasingly popular. By providing as much memory and power as required and easily scaling up or down, they handle applications with millions of users cost effectively.

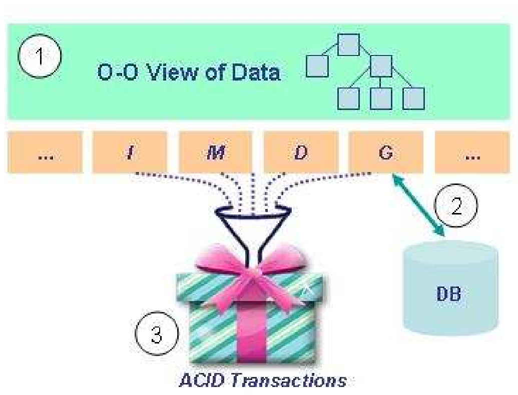
However, companies looking to run mission-critical applications in such an environment face the problem that, to date, there has been no simple, fast and robust platform to do so. This means that :



The Cloud Conundrum

- developers have to worry about distributing data across nodes, finding it again and following links from one machine to another to retrieve the data, sometimes in the face of outages;
- an In Memory Data Grid (IMDG) needs to link back to existing data sources like databases, so that feeds to data warehouse applications can be used;
- classical distributed transactions are almost never used as they are slow and unreliable, workarounds are expensive and difficult to maintain.

CloudTran Design Overview



CloudTran solves these problems through its unique design which incorporates :

1. **ORM** : the Object-Relation Mapper gives Java developers an O-O view of the data grid, converting between Java object references and in-memory "rows" with foreign keys. It also handles the distribution issues such as multi-node searches and collecting rows from different nodes into a single object graph;
2. **IMDG - DBMS Mapping** : CloudTran converts a DBMS Schema into the "row" objects (with foreign keys) that go into the IMDG, and handles initial load and object storage functions; the architecture also provides plug-in points for non-database stores
3. **ACID Transactions** : CloudTran is the first Transaction Monitor specifically designed for scalability and performance in public and private clouds. It provides high performance, rock-solid ACID guarantees and can handle any number of nodes, for data operations and messaging, e.g. on Enterprise Service Buses.

.

Java View (ORM)

CloudTran …		**Which means that …..**
Generates the Java to load, store and hold database information in the grid.	→	Reduces the effort and problems associated with manual development.
Manages the translation between foreign keys in the in-memory grid and inter-object references in Java, and the lazy loading of objects as necessary.	→	The Java developer does not have to deal with distributing or looking up data, or work with the in-memory data.
Has a very simple API.	→	There is no need to invest in lengthy training or investigation to get the job done.
Supports "Entity Groups", a technique to locate similar information in the same machine	→	Reduces cross-machine lookups and makes the application as fast as possible
Manages changes in memory objects and forwards the necessary changes to the in-memory grid	→	It is fast and easy to program
Handles distribution of data across the grid, including scalable partitioning and backups	→	CloudTran is automatically scalable, design once and deploy.

Mapping database to in-memory store

Uses in-memory data to search and read information faster than using a disk-based data store.	→	Very high levels of performance / instant response times can be achieved.
Allows entities to be grouped for performance which avoids costly cross-machine calls to gather information e.g. a customer's record could be stored in the same machine as their orders and addresses.	→	A much better application performance is achieved than using a random distribution of objects; this can often give an order-of-magnitude improvement in performance.
Generates Java code to automatically represent database rows in memory, load rows into the correct machine and store changed objects.		It is right first time, time is not wasted on manual processes and attendant problems (finger trouble) are eliminated.
Is self healing with backups for all machines and simple failover if one goes down.	→	The performance is consistent, reliable and meets SLAs.

Transaction Management

Provides ACID (Atomic, consistent, isolated and durable) transactions for applications distributed around the cloud.	→	Developers can design applications with tried and tested approaches. On-going cost and reduced quality of service from non-ACID tools are avoided.
Fast commits by recording in backed-up memory ("distributed transactions" are normally optimised for slower environments and record commits to disk).	→	Commit performance is very high (many thousands per second) leading to better user experience, and so repeat traffic and more business.
Restores in-flight transactions that are interrupted by failure of the primary to a backup machine from the central record.	→	Better performance against SLAs and lower ongoing support costs.
Buffers transactions so the application can carry on at full speed without waiting for disks and databases. This smooths spikes in demand and means that applications can continue if the database goes down.	→	Less expensive hardware and software for data bases can be used to deliver higher performance. Continuity of operation is assured.
The transaction management is universal with plug-in points for handling messages and non-RDBMS data sources all coordinated in a single transaction.	→	Reliable applications can be built with the same facilities as in J2EE.
Is scalable, the same design works from the smallest transactions to the largest.	→	There is no need to rearchitect successful applications.

For more information :
Matthew Fowler : New Technology/*enterprise* Ltd, +44 207 529 9797
http://cloudslave.blogspot.com

CloudTran V1 is offered on the GigaSpaces platform at $1,500 per deployed core for small volumes (up to 50). Free for development.