GIGASPACES
SINGLE PLATFORM. COMPLETE SCALABILITY

Social E-Commerce

An Architecture Case Study

April 2011

Introduction

In February 2011 GigaSpaces CTO Nati Shalom and Tomer Gabel, an application engineer at Delver (Sears Holding Company, Israel), met to discuss social e-commerce and the scalability challenges associated with handling social graphs.

This paper summarizes the main points that came from that discussion. The full video is available here.

Executive Summary

A case study of the architecture used by Delver/Sears, of how Sears built a social e-commerce solution that can handle complex relationship queries in real time. The case study includes the architectural considerations behind their solution , why they chose memory over disk, how they partitioned the data to gain scalability, why they chose to execute code with the data using the GigaSpaces Map/Reduce execution framework, how they integrated with Facebook, and why they chose GigaSpaces over Coherence and Terracotta.

What is Social E-Commerce?

According to Wikipedia, Social Shopping is an e-commerce mechanism where shoppers' friends become involved in the shopping experience. Social shopping attempts to use technology to mimic the social interactions found in physical malls and stores.

The Opportunity

A recent study showed that over 92 percent of executives from leading retailers are focusing their marketing efforts on Facebook and related applications. Furthermore, over 71 percent of users have confirmed they are more likely to make a purchase after "liking" a brand they find online. (source)

Subscribers, Fans, & Followers: The Social Break-Up is a report that analyzed social behavior, and more specifically why consumers end brand relationships. Though ending of brand relationships is not the focus of this paper, some of the statistics in the report are useful for quantifying where we are today with social e-commerce in terms of the market size and its potential.

Facebook statistics:

- 73% of US online consumers have a profile on Facebook.
- 65% of US online consumers are currently active on Facebook.
- 42% of US online consumers (64% of those on Facebook) have "liked" a company on Facebook.

Twitter statistics:

- 17% of US online consumers have a Twitter account.
- 9% of US online consumers are currently active on Twitter.
- 5% of US online consumers (56% of those on Twitter) are "followers" of at least one company
- 71% of FOLLOWERS expect to receive marketing messages from companies through Twitter.

Clearly the adoption of social commence is quite staggering, representing unprecedented opportunities for online retailers. And yet, when growth happens that rapidly, there are usually technological barriers along the way.

This paper's focus is on one specific aspect of the technology challenges associated with building a social e-commerce platform – Scalability and the "Social Graph."

The Social Graph Challenge

The Social Graph is "the global mapping of everybody and how they're related."

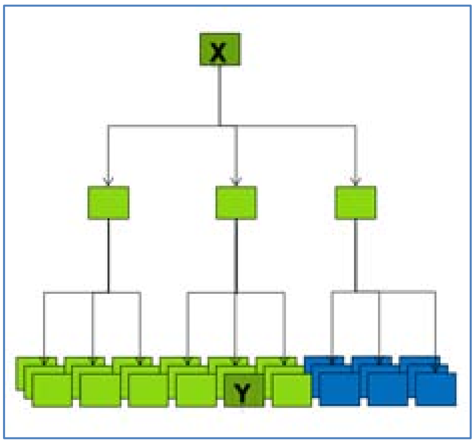
Processing social networks is not an easy proposition:

- Massive amounts of branching data
- No data locality
- Very few assumptions can be made about the data

In other words, to meet the capacity scaling demand, you cannot store the data in a centralized location. That forces you to distribute the data. On the other hand, to access (or query) the data, you cannot assume that the data for the query is located in a single node. That forces you to look for the data on multiple nodes.

Let's take a simple Facebook-based scenario to get some sense of the complexity of the problem:

- Imagine every Facebook user (500 million)
- Imagine each person is only connected to 100 others (conservative estimate)



The query: How is user X connected with Y?

- X has 100 friends
- Each of them has 100 friends
- 10,001 nodes visited!
- 101 reads from the underlying storage system!

A database can only sustain a given number of queries at a given time. Multiply this scenario by all the users in the system at the relevant time... Clearly, even if we can scale at the capacity level, we cannot scale at the query level.

The crux of the problem:

- High branch factor necessitates many loads to serve even a simple request
- No data locality + high branch factor means highly random I/O
- Traditional storage models (RDBMS, flat files, etc...) are a poor fit. A storage model is needed that can save complete object graphs.
- Building a graph that traverses partitions is practically impossible in terms of performance.

The Solution

Sears/Delver chose the following architecture elements to resolve the social graph challenge:

- **Use memory as the main storage**
  Random I/O access works much better on memory devices than on disk (see more details here)
- **Execute the code with the data, using real-time Map/Reduce**
  To reduce the number of iterations required to execute a particular query, we use the executor API. The executor API enables us to push the code to the data. In this way, we can execute complex data processing on the data node at memory speed vs. network speed.
- **De-normalize the data**
  To reduce the amount of traversal access and network hops per query on the graph, we need to copy elements of the graph into each node. For example, the list of friends and friends of friends (up to a specified degree) could be stored in each node, and thus become available to any element of the graph without the need to consult with other nodes.

The Operational Perspective

Scaling social e-commerce involves not only application architecture but also operational aspects. This is important because e-commerce and social sites tend to evolve rapidly, and therefore the time it takes to release a new feature from development to production is critical. This process is often referred to as *continuous deployment*, or in our specific case it can also be referred to as *continuous scaling*.

There are two factors in this solution that are important to achieve continuous scaling/deployment:

- **Automation:** If the process of deployment and scaling involves a lot of human intervention, then the time it takes to release a new feature becomes significantly higher. It is therefore important that the entire process can be fully automated. In some cases, it may be desirable to have manual checkpoints in the process, but even in those cases the expectation is that manual intervention would involve clicking on a <continue> button and nothing more. To achieve this level of automation, we need to interact with the application infrastructure through an API. This process of automation and integration between the development and operational environment is often referred to as DevOps. The GigaSpaces reference to a DevOps API is provided here.
- **Schema evolution:** In many cases we might want to add or change an application's data structure as part of the upgrade process, without bringing the system down. This is considered one of the more complex challenges with many existing databases. A document model is schema-less and therefore more suited for this sort of rapid data change.

U.S. Headquarters
GigaSpaces Technologies Inc.
317 Madison Ave, Suite 823
New York, NY 10017
Tel: 646-421-2830
Fax: 646-421-2859

U.S. West Coast Office
GigaSpaces Technologies Inc.
101 Metro Drive, Suite 350
San Jose, CA 95110
Tel: 408-878-6982
Fax: 408-878-6149

International Office
GigaSpaces Technologies Ltd.
4 Maskit St., P.O. Box 4063
Herzliya 46140, Israel
Tel: +972-9-952-6751
Fax: +972-9-956-4410

Europe Office
GigaSpaces Technologies Ltd.
2 Sheraton St.
London, W1F 8BH, United Kingdom
Tel: +44-207-117-0213
Fax: +44-870-383-5135