SEVENTH FRAMEWORK PROGRAMME

FP7-ICT-2011-1.5 Networked Media and Search Systems
b) End-to-end Immersive and Interactive Media Technologies

Specific Targeted Research Project

VENTURI

(FP7-288238)

immersiVe ENhancemenT of User-woRld Interactions

D2.6.1 First server-side API and architecture

Due date of deliverable: [31-07-2012]
Actual submission date: [02-08-2012]

Start date of project: 01-10-2011
Duration: 36 months

1.1.1.1 Summary of the document

Document Code:	D2.6.1 First server-side API and architecture –v1.0
Last modification: State:	02/08/2012 Ready for submission
Participant Partner(s): Editor & Authors (alphabetically): Fragment:	metaio Editor: Selim BenHimane, Paul Chippendale (FBK) Author: Selim BenHimane (metaio), Tania Farinella (metaio) No
Audience:	☒ public ☐ restricted ☐ internal
Abstract:	This document is deliverable D2.6.1 “First server-side API and architecture” and presents the improvements done in the server structure and the different optimizations.
Keywords:	AR framework, AR SDK, server-side API
References:	Refer to the corresponding section at the end of the deliverable

1.1.1.2 Document Control Page

Version number	V1.0
Date	31/07/2012
Modified by	Selim BenHimane, Tania Farinalla
Comments	First version of the document
Status	☐ draft ☒ WP leader accepted ☒ Technical coordinator accepted ☒ Project coordinator accepted
Action requested	☐ to be revised by partners involved in the preparation of the deliverable ☐ for approval of the WP leader ☐ for approval of the technical coordinator ☐ for approval of the project coordinator

1.1.1.3 Change history

Version number	***Date***	***Changed by***	***Changes made***
1.0	31/07/2012	S. BenHimane, Tania Farinella	Initial version from template, ToC, summary.
1.1	02/08/2012	Paul Chippendale	Final check

Table of Contents

1 Executive Summary 5
1.1 Audience 5
1.2 Summary 5
1.3 Structure 5
2 Introduction 5
3 Released software 5
3.1 SOLR integration in the junaio server 5
3.2 Amazon Route 53 6
3.3 Amazon Auto Scaling 6
3.4 Local Geo-localization 6
3.5 Caching-based optimization 6
4 Conclusions 6
5 References 7

1 Executive Summary

1.1 Audience

This deliverable is public software.

1.2 Summary

The document briefly describes what has been released for the first server-side API and architecture.

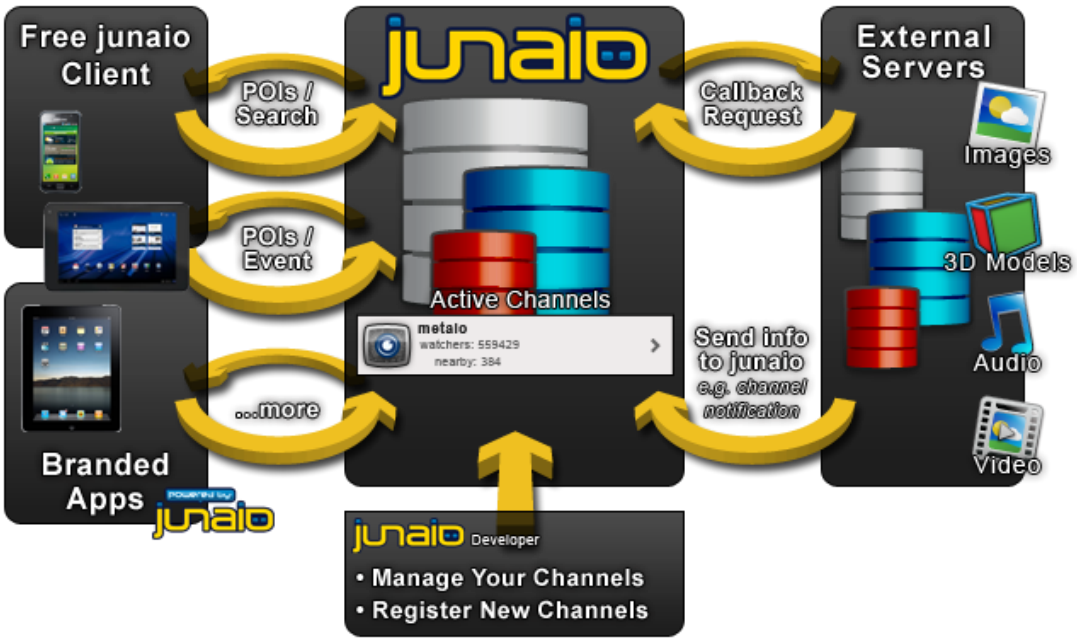
1.3 Structure

This deliverable is structured as follows: Section 2 contains an introduction to the report. Section 3 describes the different improvements done for the VENTURI first year server-side API. Sections 4 and 5 conclude the document with future work and references to available web pointers.

2 Introduction

Objective: Implementation of the digital media server and server-side services.

Tasks: Development of an open server API for hyper-media access based on the current junaio API [1], enabling the integration of static and dynamic hypermedia objects or external sources like Google Maps etc. Development of an optimization of the server structure processing (intelligent caching, fast retrieval…) to ensure a good user experience.



3 Released software

3.1 SOLR integration in the junaio server

In the context of server structure processing optimization, we tested replacing part of the engine responsible for the Point of Interest (POIs) database management which used to be based on Oracle with a database management based on Solr. Solr includes a powerful full-text search; this will speed up the POIs full text search. POIs text information is preprocessed and stored into indexes that enable a fast retrieval of the interesting POIs.

3.2 Amazon Route 53

We tested the usage of Amazon Route 53 [3] for faster requests responses (Geo DNS) which allows queries for a domain to be automatically routed to the nearest DNS server (US-East, US-West, Asia, Europe). Servers on the Amazon Cloud [2] are organized in regions: Europe (Ireland), US-West-1 (California), US-West-2(Oregon), US-East (Virginia), Asia-1 (Tokyo), Asia-2 (Singapore), South America (San Paulo). Several servers have been started in Europe (Ireland), US-West-1 (California), US-East (Virginia), and Asia-1 (Tokyo). In each single region, requests are routed by a Load Balancer towards the server having the lowest load. Amazon Route 53 performs a latency-based routing and forwards the requests to the nearest Load-Balancer.

3.3 Amazon Auto Scaling

We tested the usage of Amazon Auto Scaling [4] service to automatically start new servers when any server is not responding anymore, to automatically start extra-servers when the existing ones are overloaded and to automatically stop extra-servers when the load returns to standard levels. The system is now able to automatically react in a time-window going from 20 minutes in Europe to 45 minutes in other regions to variations of load.

3.4 Local Geo-localization

We tested the usage of a local PostgresSQL[5] database on each server to perform the geo-localization of users. Given the position of the user (latitude, altitude, longitude) the region in which he is located is retrieved, accessing a local database instead of calling the Oracle database that is located on a different server. The geo-localization is now performed in 70ms on each server, while before it required from 500ms to 680ms.

3.5 Caching-based optimization

We experienced a better caching in the server to reduce the communications between the database and the servers. Basic data about Junaio channels are stored in a cache and refreshed every five minutes. The refresh is now synchronized to user requests in order to not introduce any delay. Results of database requests are stored in a cache for one hour. This reduces the number of database calls during the usage of Junaio by a single user and when multiple users are accessing Junaio from the same area.

As the database is located in Europe, each call to the database needs from 30ms (Europe) to 180ms (other regions). Every request to Junaio can require from 1 to 16 database calls. The use of the cache enable us to provide response time from 30ms to almost 3 seconds faster.

We also tested better caching methods, this has allowed us to modify the system in order to supply basic Junaio functionalities (most popular channels, newest channels, favorite channels, history, channel search, POIs search, visual search) even when the database is not reachable.

4 Conclusions

Thanks to the new improvements, the system is able to provide the user with faster responses independently from the location in which Junaio is accessed.

The system automatically scales when the user demand changes. The system keeps on working even when some components (servers or database) are not available.

The system will be improved introducing a replication of the database in different regions in order to reduce the access latency and introducing content distribution.

5 References

[1] http://www.junaio.com
[2] http://aws.amazon.com/ec2/
[3] http://aws.amazon.com/route53/
[4] http://aws.amazon.com/autoscaling/
[5] http://www.postgresql.org/