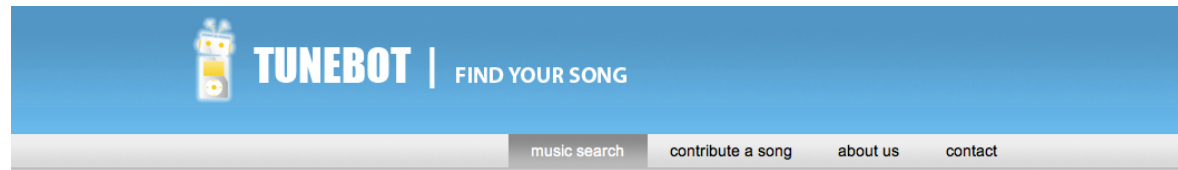


Tunebot in the Cloud

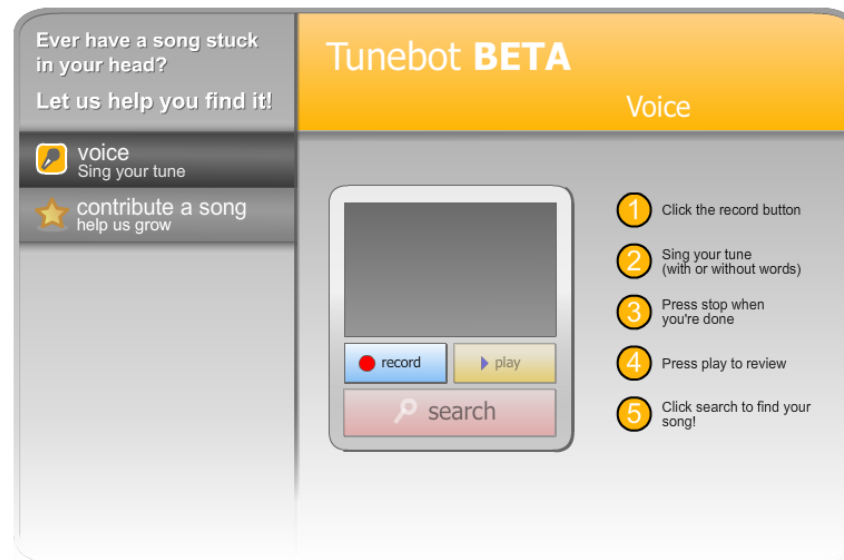
Arefin Huq

18 Mar 2010

What is Tunebot?



iTunes has over 3 Million songs in its Database. We only have 3052.
Help us grow by [Contributing A Song](#).



The interface is titled "Tunebot BETA" and "Voice". It is divided into a left sidebar and a main content area. The sidebar contains the text "Ever have a song stuck in your head? Let us help you find it!" and two options: "voice Sing your tune" (with a microphone icon) and "contribute a song help us grow" (with a star icon). The main content area features a central control panel with a "record" button (red circle), a "play" button (yellow triangle), and a "search" button (magnifying glass). To the right of the control panel is a numbered list of five steps: 1. Click the record button, 2. Sing your tune (with or without words), 3. Press stop when you're done, 4. Press play to review, and 5. Click search to find your song!



What is Tunebot?

<http://tunebot.cs.northwestern.edu>

- Automated online music search engine for query-by-humming (QBH).

What is Tunebot?

<http://tunebot.cs.northwestern.edu>

- Automated online music search engine for query-by-humming (QBH).
- Users sing or hum tunes to search.
- Queries are matched against other sung examples that have been contributed.

What is Tunebot?

<http://tunebot.cs.northwestern.edu>

- Automated online music search engine for query-by-humming (QBH).
- Users sing or hum tunes to search.
- Queries are matched against other sung examples that have been contributed.
- Current DB: nearly 10K examples of over 3000 songs

What is Tunebot?

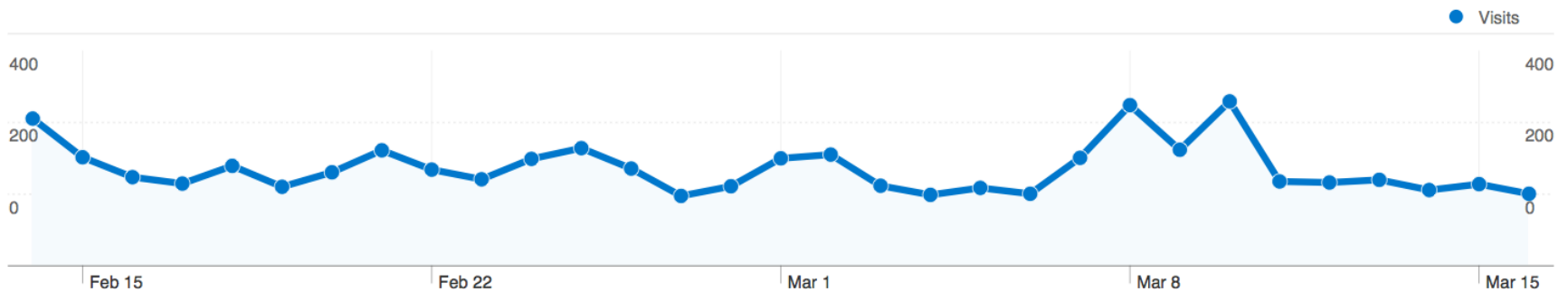
- A project of the Interactive Audio Lab led by Prof. Bryan Pardo (EECS)
<http://music.cs.northwestern.edu>
- Single-machine locally-hosted installation
- Web/Flash/iPhone client
- PHP/MySQL server-side front-end on Apache
- Java/MySQL back-end as a Tomcat servlet

Tunebot Traffic

(Source: Google Analytics)


tunebot.cs.northwestern.edu
Dashboard

Feb 14, 2010 - Mar 16, 2010
Comparing to: Site




Site Usage


 **5,621** Visits

 **86.41%** Bounce Rate

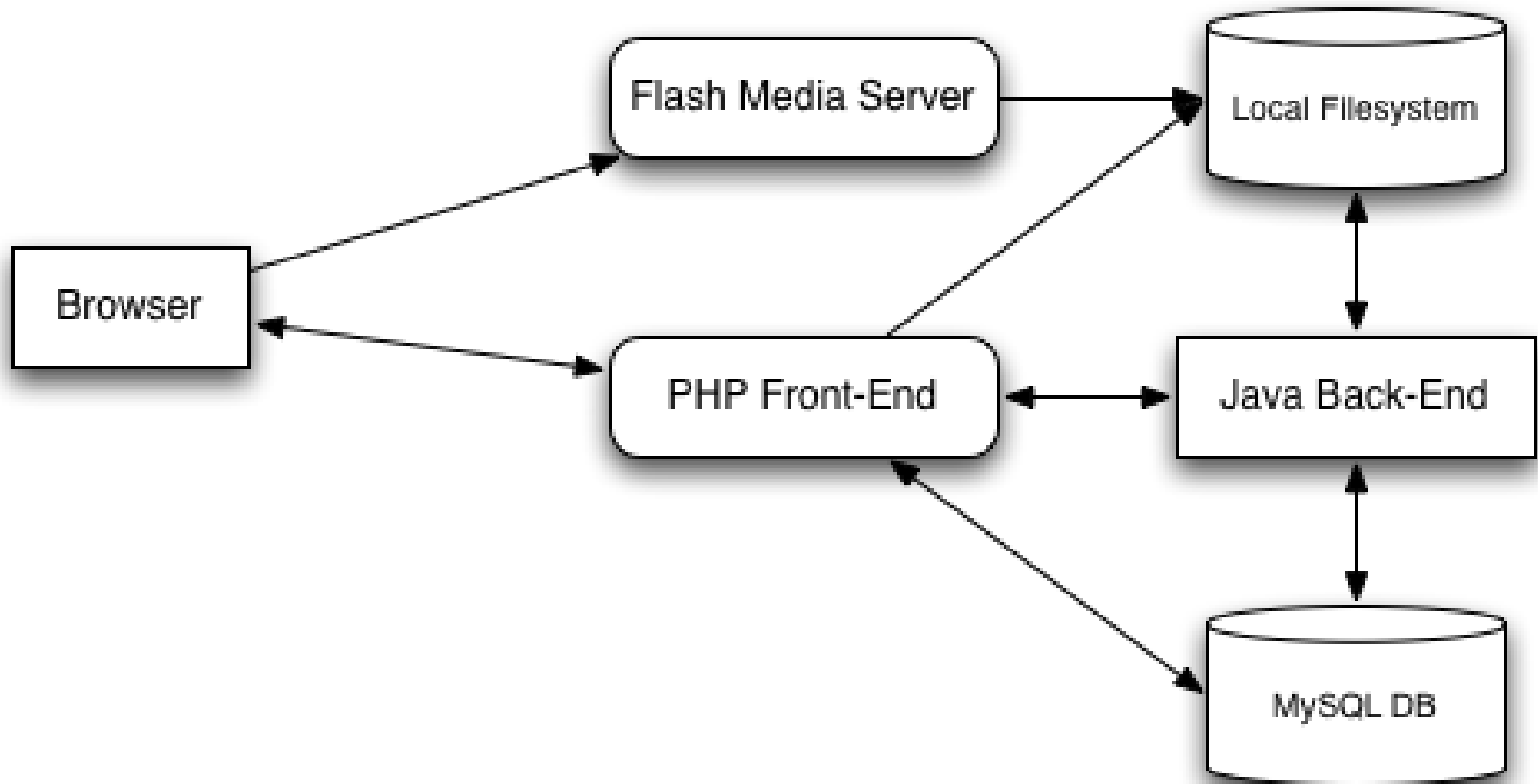
 **8,100** Pageviews

 **00:01:18** Avg. Time on Site

 **1.44** Pages/Visit

 **93.42%** % New Visits

Architecture



Goals

- Improve query response time to < 1 sec
 - Typical query takes 5 seconds to complete
 - Computation is linear in DB size

Goals

- Improve query response time to < 1 sec
 - Typical query takes 5 seconds to complete
 - Computation is linear in DB size
- Handle larger database
 - DB expected to grow to “critical mass” of 10K songs

Goals

- Improve query response time to < 1 sec
 - Typical query takes 5 seconds to complete
 - Computation is linear in DB size
- Handle larger database
 - DB expected to grow to “critical mass” of 10K songs
- Adapt to growing and varying load
 - Handle traffic spikes

Project Status

Task
Port front-end to Linux
Port back-end to Linux
Write load-testing framework
Deploy back-end in Cloud (single instance)
Deploy front-end in Cloud (single instance)
Decouple database from front-end
Deploy load-balancing front-end with replicated DBs
Parallelize search for single user
Dynamic provisioning

Project Status

Task	Done?
Port front-end to Linux	Yes
Port back-end to Linux	Yes
Write load-testing framework	No
Deploy back-end in Cloud (single instance)	Yes
Deploy front-end in Cloud (single instance)	Yes
Decouple database from front-end	No
Deploy load-balancing front-end with replicated DBs	No
Parallelize search for single user	Yes
Dynamic provisioning	No

Linux Port

Linux Port

- equivalent software installation and configuration
- file system dependencies
- permissions
 - groups, sticky bits, umask
- Adobe Flash Media Server

Amazon Cloud

Amazon Cloud

<http://ec2-174-129-48-175.compute-1.amazonaws.com/beta/>

- Ubuntu 9.10 64-bit AMI
- EC2 m1.large instance
 - 2 virtual cores x 2 EC2 compute units
 - 7.5 GB memory
 - “high” IO
- Elastic Block Store

Parallelization

Algorithm

- Users contribute songs to the database by singing a portion of the melody and labeling it.
- This is converted to an internal representation (NIS) that is similar to MIDI or musical notation. This representation is the database key.
- When a user searches for a song their singing is also converted to NIS format.

Parallelization

- Search requires a linear scan of the database and a score of how well the query matches each key.
- Overall algorithm is $O(n)$.
- Cost of computing each score is proportional to st where s and t are the lengths of the query and key respectively.
- Solution is to compute scores in parallel.

Except...

That's not the problem.

Except...

That's not the problem.

Most of the time is spent converting the input audio file to NIS format.

(all times in seconds)

Query Length	Query response time (HttpServletRequest:doPost)	Audio conversion time
0.19	0.147	0.024
18.11	4.99	4.58
48.51	13.2	12.4

Speeding up Audio Conversion

- Replace hand-coded conversion with optimized third-party code.
- Eliminate lots of unnecessary data copying.
- Improve parameter choices?
- Use `System.arrayCopy()` instead of for-loops.
- Convert audio while streaming?
- Parallelize the audio conversion?

Parallelization

But $O(n)$ will be an issue eventually.

Parallelization

But $O(n)$ will be an issue eventually.

(Intel 2.4GHz Core 2 Quad 64-bit Java)

Query Length	1 thread	4 threads	Speedup
0.19 s	40 ms	23 ms	1.7x
18.11 s	331 ms	195 ms	1.7x
48.51 s	749 ms	425 ms	1.8x

Database and File System

Database and File System

- Front-end and back-end talk directly to the database and file system. Back-end assumes all files (currently 30+ GB) are available.
- Requires architectural redesign to change.
- Amazon Cloud does not have a notion of shared file system.
 - Instance Store
 - EBS
 - Simple Storage Service

Database and File System

- Database usage pattern is simple – mostly SELECT with few predicates or joins.
- Amazon Simple DB
- Amazon Relational Database Service

Dynamic Provisioning

- Amazon CloudWatch
- Amazon Elastic Load Balancing
- Amazon Auto Scaling

All depends on decoupling database and file system and managing concurrency.

Need to remove hardcoded URLs in PHP and Flash.