TiVo

HAXE IN THE ENTERPRISE: A CASE STUDY

Alfred Barberena

Who is TiVo? What do we do?

TiVo provides advanced television to the cable industry in the US and abroad

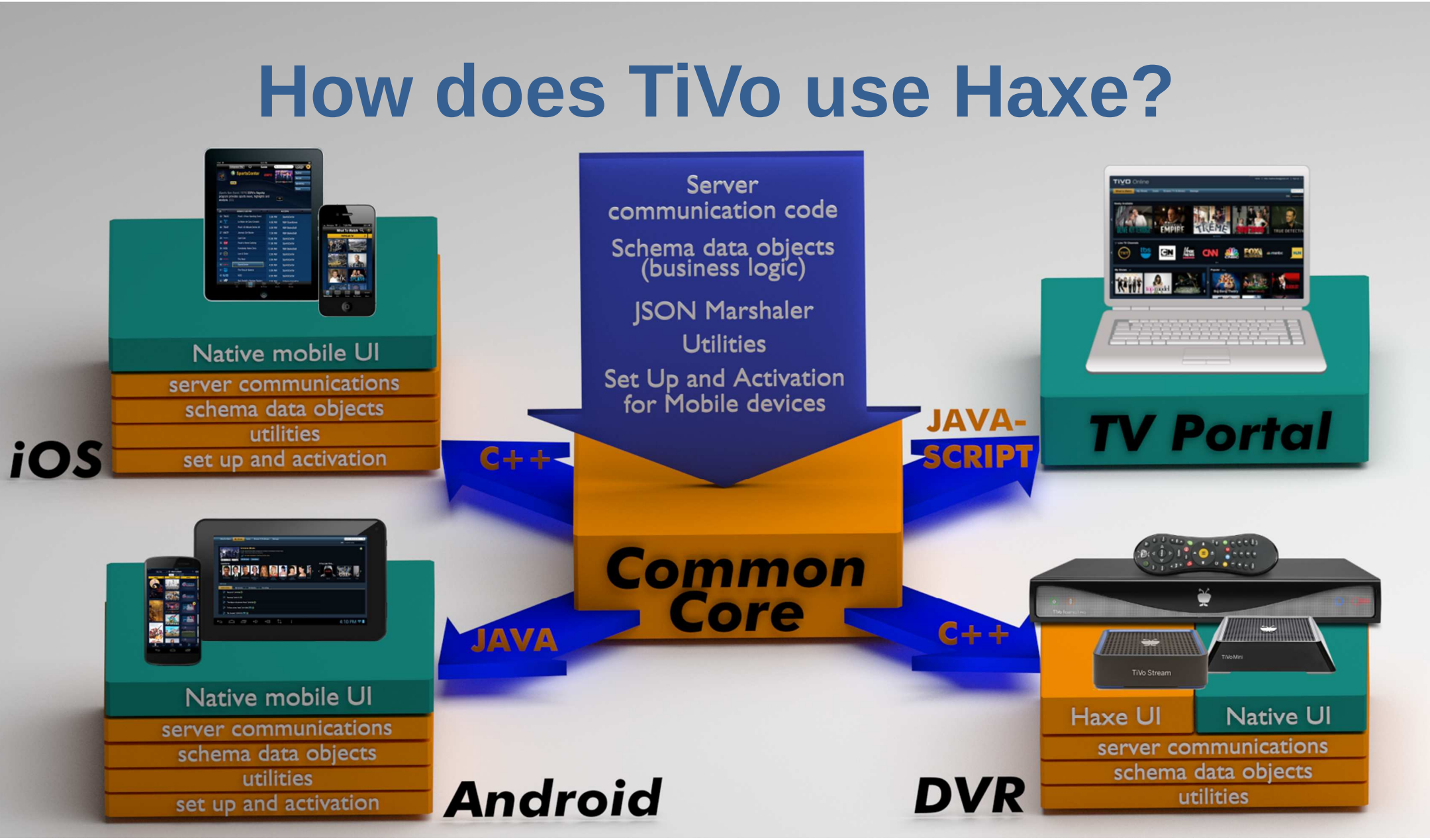
5.5 Million active subscribers

- US comcast
- Canada COGECO
- Sweden com hem
- Spain ONO
- UK Virgin media

- Digital Video Records
- Second Room Video Devices
- Mobile Applications
- Online Web Video Portal


TiVo Portfolio
TiVo Data Services
0% Haxe Next Target
TV Portal
70% Haxe UI Code 500 KLOC
60% Haxe 250 KLOC
TiVo Stream
TiVo Mini
DVR
Mobile

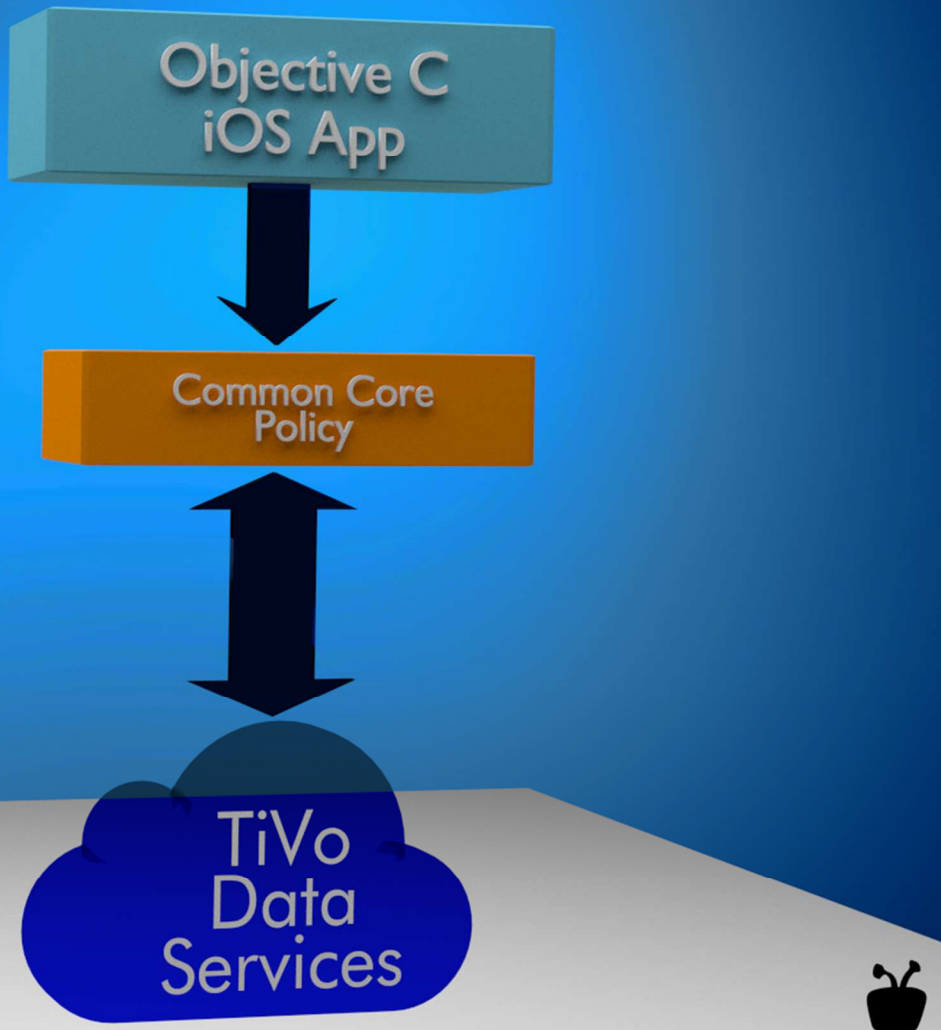
How does TiVo use Haxe?



How does TiVo use Haxe? Call in

How does TiVo use Haxe? Call in

Performance gain over initial Objective C implementation



Why did TiVo move to Haxe?

- Initially TiVo used Adobe Flash for embedded devices

  Pros:

  - ActionScript 3 language with a large pool of developers
  - Ample developer tools available

  Cons:

  - Application performance poor on low powered devices
  - JIT compiler slowed initial screen entry performance
  - Difficult to improve Flash engine

Why did TiVo move to Haxe?

Before Haxe

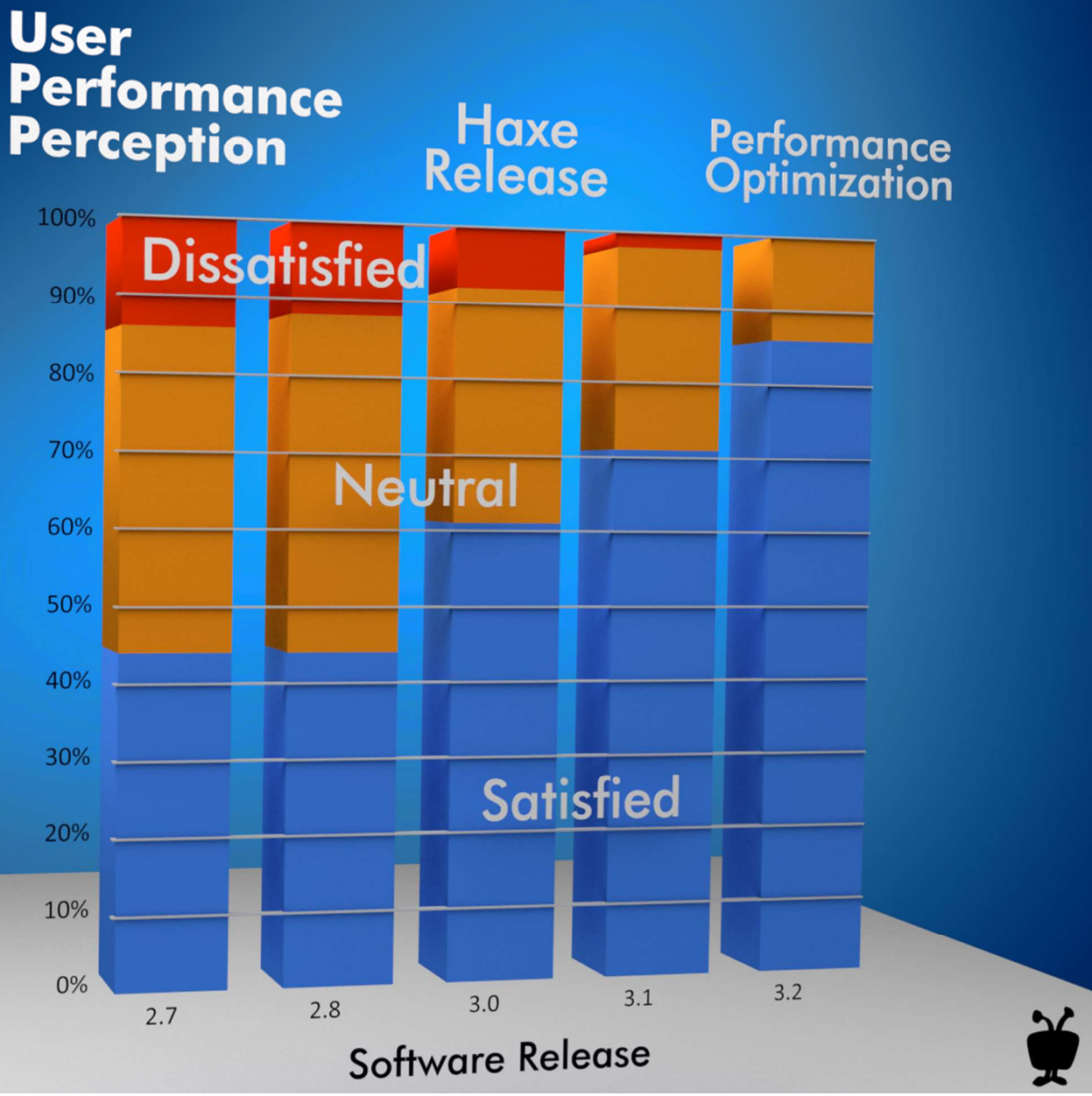
“Very sluggish UI”

After Haxe

Performance Optimizations

0% Dissatisfied

on low powered DVRs



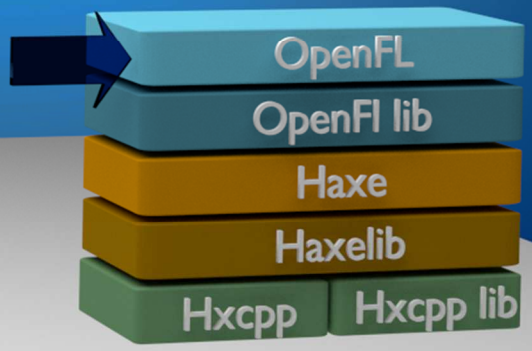
Challenges: Optimize Performance

- Analyze Haxe Performance
- Identified two methods
  - Quantified object churn
    - Track the number of objects created for a specific UI actions and who created the objects
  - Functional call frequency
    - Used process logging at kernel level (plog) to track function call frequency and rank code areas by the total number calls during UI functionality

Optimize Performance

Eliminated unused events in OpenFL – “API cheating”

- Removed ADDED and REMOVED events
- Removed the “capture” phase of key events
- ENTER_FRAME event was changed to a singleton and dispatched only to the stage
- Removed 60K out of 70K (84%) events during typical UI operation



Optimize Performance

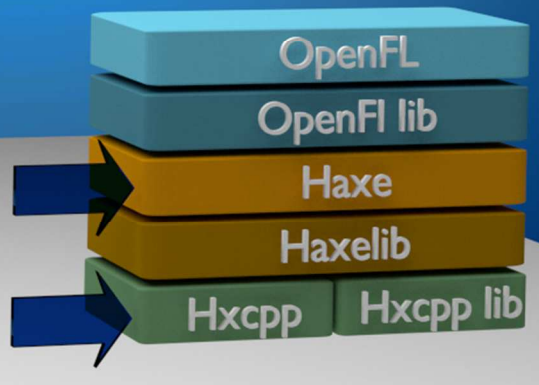
Implemented a faster version of Socket.select

- Does not require the creation of multiple Array classes on every call
- Produces zero object churn
- Change to hxcpp – submitted back to open source
- Identified through object churn analysis

Optimize Performance

Replaced std::map with std::tr1::unordered_map:

- Hash table with O(1) complexity
- Faster in parts of the UI
- Change to hxcpp lib – reject for a different hash table implementation
- Identified through "*function call frequency*" analysis



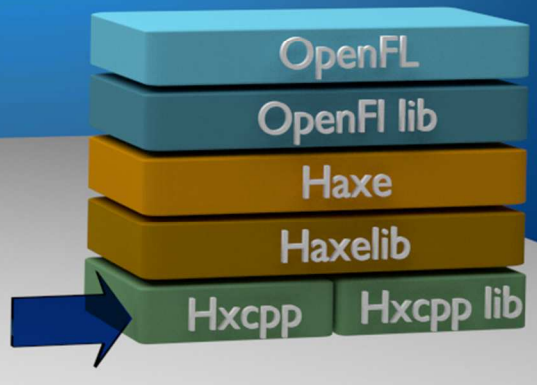
Optimize Performance

Replaced garbage collector with a version which supported Big/Little Endian

- Measurably faster within the TiVo UI app (7x)
- No pull request because API is different for current stock hxcpp GC

Optimization of JSON parsing URL decode

- Change to hxcpp – submitted back to open source
- Identified through object churn analysis



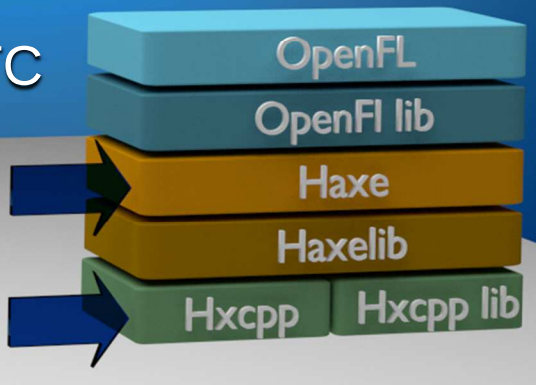
Obstacles Overcome

UTF8 string deficiency

- Required an additional UTF8 handling class to support Swedish characters
- UTF8 string support in Haxe standard libraries continues to be poor

Additional Date/Time support

- Support DST, offset awareness
- Support for UTC time parts
- Extended toString/fromString to handle time zones, UTC



Obstacles Overcome

Building from source

- Required to build from source for MIPS, MIPSEL, ARM flavors
- Build file for hxcpp

Full feature IDEA for Haxe

- Required to improve IntelliJ IDEA Haxe plugin to meeting developers requests

Future Direction

- Overall the migration to Haxe has been a tremendous win
- Extending the longevity of low powered embedded devices
- Enabled code sharing across multiple platforms across the enterprise
- As a future direction plan to expand the use of Haxe across the TiVo Enterprise