Cedric BAIL

Samsung Research America

Tizen Native UI

A True & Free Software Graphical Toolkit Designed for the Embedded World

A True & Free Software Graphical Toolkit Designed for the Embedded World

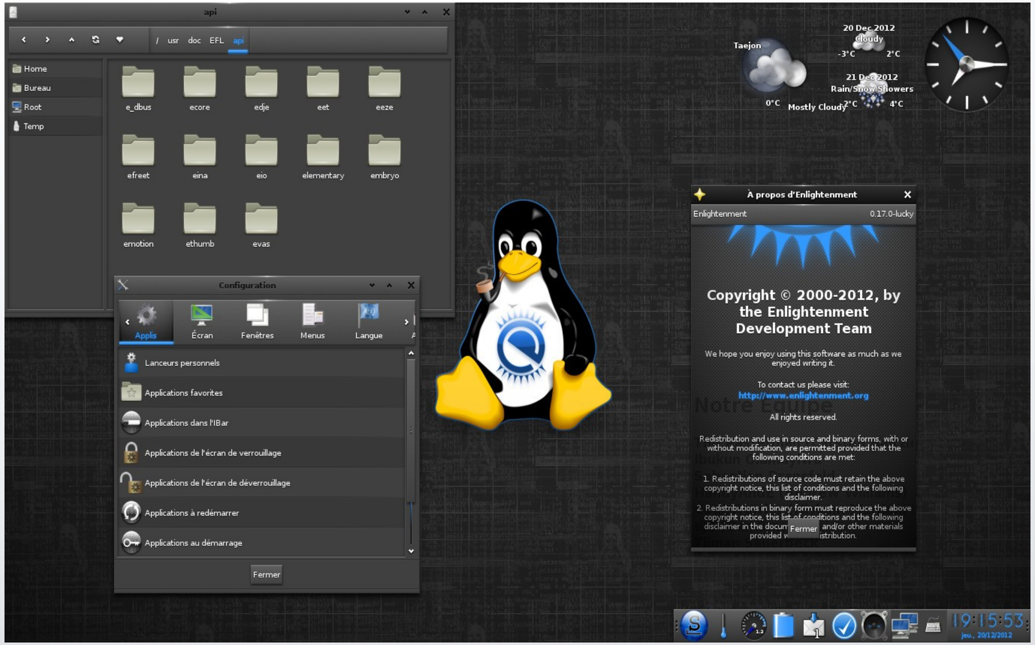
- Introduction to EFL history
- Optimization for the Embedded World
- Current Work
- Possible Areas of Improvement

Who am I?

- Cedric BAIL <cedric@osg.samsung.com>
- Working on embedded technology since 2004 (mobile, set top box, …)
- Working on Enlightenment technology since 2007
- Working for Samsung since 2011
- Gained some experience along the way on:
  - Optimization (CPU, memory, battery)
  - Rendering pipeline

Tizen Native UI – EFL, Enlightenment Foundation Libraries

- What are the Enlightenment Foundation Libraries?
  - Toolkit created for Enlightenment 17



À propos d'Enlightenment

Enlightenment 0.17.0-lucky

Copyright © 2000-2012, by the Enlightenment Development Team

We hope you enjoy using this software as much as we enjoyed writing it.

To contact us please visit:
http://www.enlightenment.org

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the docum Fermer and/or other materials provided w istribution.

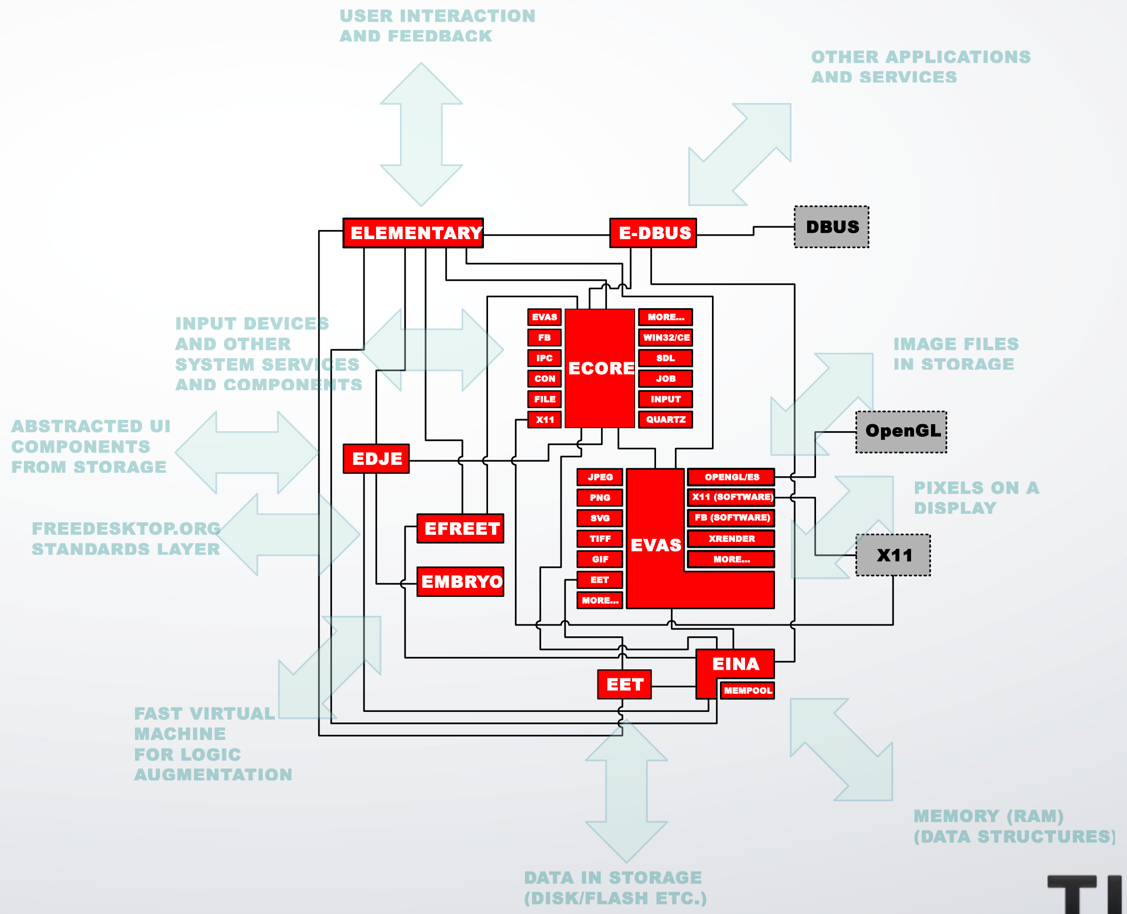
Tizen Native UI – Enlightenment 17

- Enlightenment project started in 1997 as a window manager
- First window manager of GNOME
- Full rewrite started in 2001, same time EFL development started
- Main belief was that there will never be *“a year of the Linux desktop”*
- Enlightenment is first trying to serve its developer base
- Enlightenment is usable in embedded devices
- Tizen uses it for Window Management
- Fills need for a toolkit that scales from embedded to high end desktops
- Fills need for a stack that will serve multiple applications on embedded devices

Tizen Native UI – EFL, Enlightenment Foundation Libraries

- GUI toolkit that targets embedded devices
- Licensed under a mix of LGPL and BSD license
- Optimized to reduce CPU, GPU, memory and battery usage
- Supports international language requirement (LTR/RTL, UTF8)
- Supports all variations of screen and input device (scale factor)
- Full themability (layout of the application included)
- Profile support
- Could be made to fit in 8MB with a minimal set of dependencies included
- Modular design

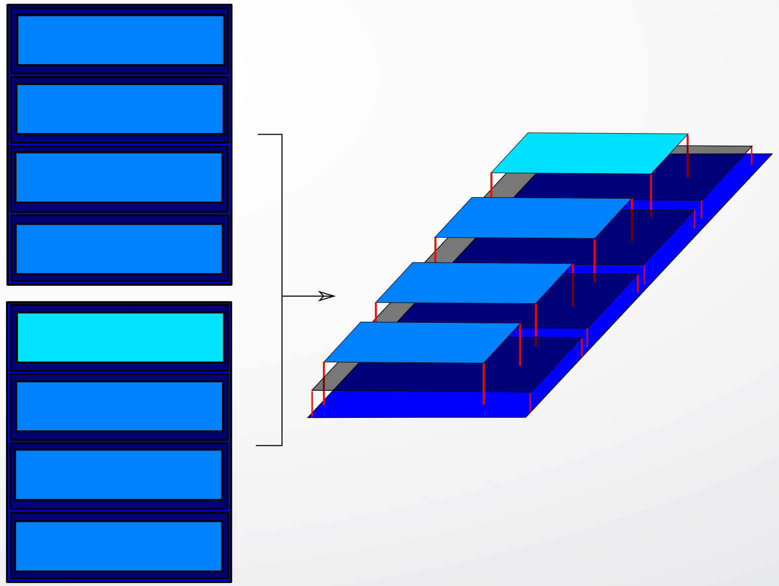
Tizen Native UI – EFL, Enlightenment Foundation Libraries


USER INTERACTION AND FEEDBACK
OTHER APPLICATIONS AND SERVICES
ELEMENTARY
E-DBUS
DBUS
INPUT DEVICES AND OTHER SYSTEM SERVICES AND COMPONENTS
EVAS
FB
IPC
CON
FILE
X11
ECORE
MORE...
WIN32/CE
SDL
JOB
INPUT
QUARTZ
IMAGE FILES IN STORAGE
OpenGL
ABSTRACTED UI COMPONENTS FROM STORAGE
EDJE
JPEG
PNG
SVG
TIFF
GIF
EET
MORE...
EVAS
OPENGL/ES
X11 (SOFTWARE)
FB (SOFTWARE)
XRENDER
MORE...
PIXELS ON A DISPLAY
X11
FREEDESKTOP.ORG STANDARDS LAYER
EFREET
EMBRYO
EINA
MEMPOOL
EET
FAST VIRTUAL MACHINE FOR LOGIC AUGMENTATION
MEMORY (RAM) (DATA STRUCTURES)
DATA IN STORAGE (DISK/FLASH ETC.)

Core Differences – Evas, Scene Graph!

- The brain of EFL
- Scene graph library with more than 10 years of optimization in it
- Glitch-free rendering
- Reduces overdrawing
- Reduces memory waste by deduplicating as much as possible
- Compressed glyph rendering
- Portable (SDL, X11, Wayland, FB, DRM, Windows, Mac OS X, …)
- Optimized software renderer (MMX, SSE*, Neon)
- Optimized use of GPU (optional)
  - Support for partial updates in cases where drivers do as well
  - Reduced context and texture switch as much as possible
  - Reduced memory overhead

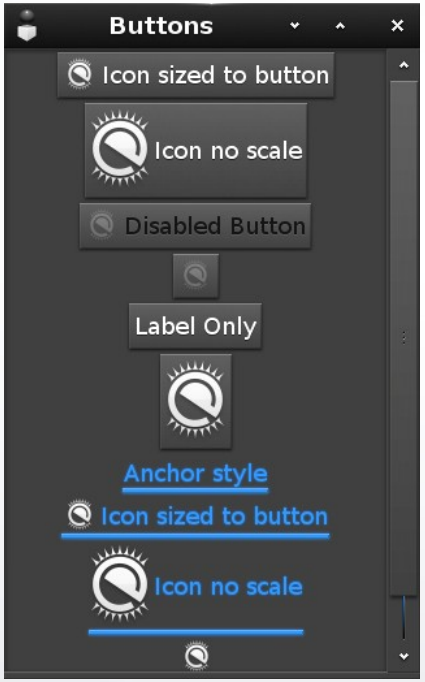
Core Differences – Evas, Scene Graph!

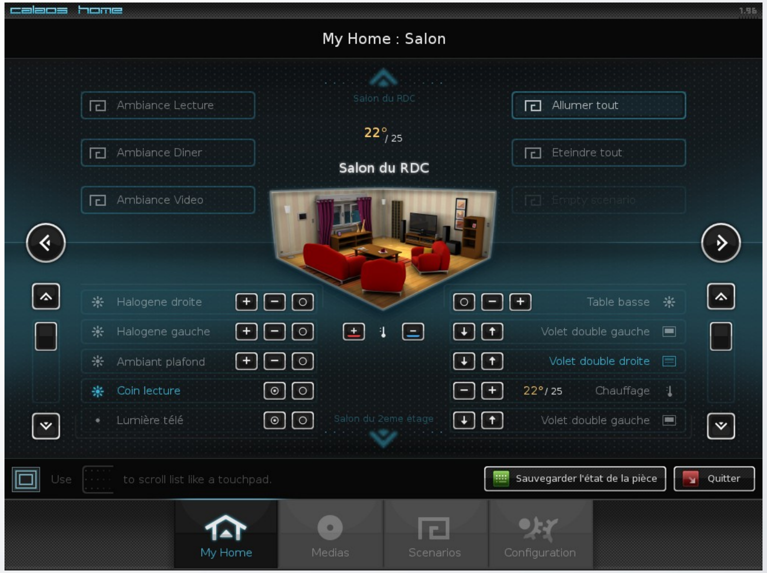


Core Differences – Edje, Theming

- The heart of EFL
- Theme and layout engine
- Descriptive langage
- Uses Evas for rendering logic (fully independent from the system)
- Doesn't require an FPU
- Optimized load time (time to first frame) and run time
- Reduced memory fragmentation

Core Differences – Edje, Theming





Core Differences– Eet, Structure Serialization

- Very specific to EFL
- Fast serialization library for file storage and network communication
- Stores image, sounds, font
- Reduces overhead to load the same data across multiple applications
- Provides tools to convert to and from a human readable form
- Configuration and theme are done using this library
- If you write C, you want Eet!

Core Differences – Elementary, Widget

- The face of EFL
- Widgets toolkit
- Use Edje and Evas infrastructure
- Screen and input independence achieved by:
  - Scale factor
  - Finger size
- Profile support (define configuration on a per-window basis)
- Fully themable
- Supports touchscreen

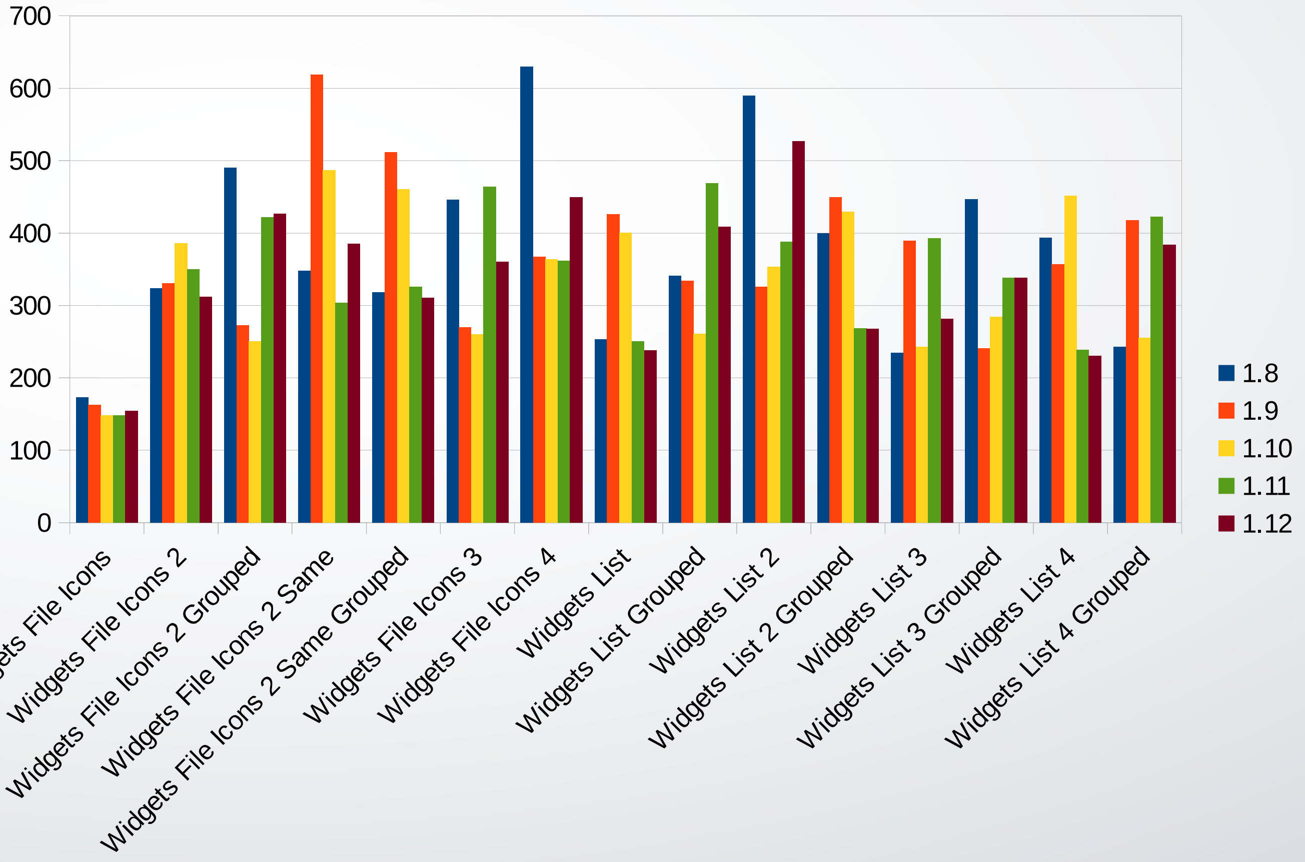
Optimization

- Before optimizing anything, you need numbers!
- Any benchmark is a partial view of an application
- Focus on revealing the biggest cost of all applications
- It will never be a complete, perfect picture
- You need reference points that are valid from one test to another

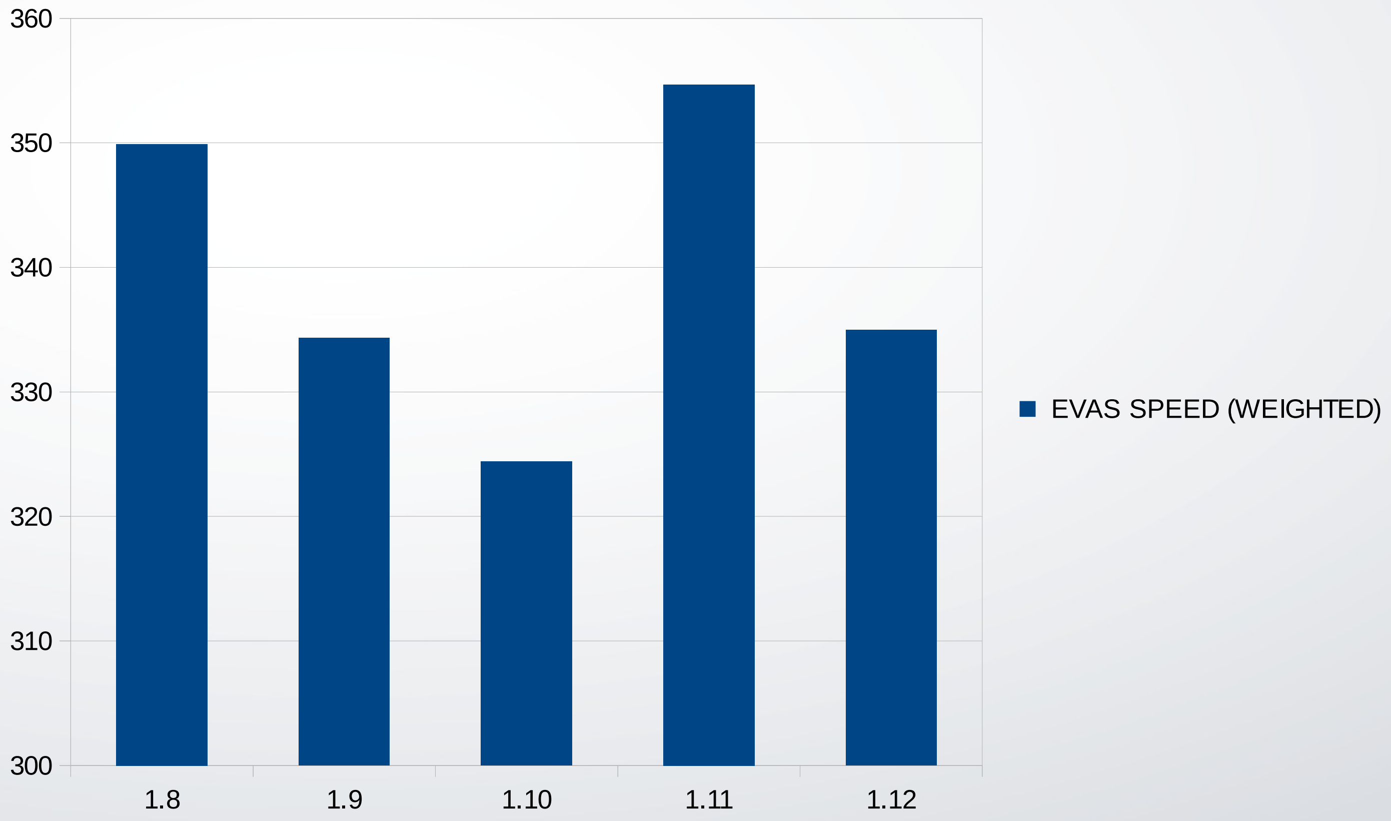
Optimization – Expedite, our Benchmark Tool

- An EFL application that forces maximum frames on screen
- Currently tests only Evas primitives directly
- Works with all previous EFL versions
- Tests specific paths in the rendering pipeline
- If EFL supports a target, expedite will work
- Provides a picture of how powerful the hardware is
- Run for every release to ensure there are no speed regressions

Optimization – Reading Result



Optimization – Reading Result



Optimization – Being Humble!

- Benchmarking reliably is difficult, especially on synthetic tests
- Even with multiple rounds
- Requires human intervention to understand what is going on
- Gives an overall idea of what is going on
- Need more complex code
- Need improved logic to determine if measures are correct

Optimization – Impact of Preloading and Shared Cache Infrastructure

- EFL is designed for the embedded world, but is used on the desktop as well
- Multiple applications running at the same time
- The more applications capable of runing at the same time, the better
- Time to first frame is critical
- Scenarios with no swap is common

Optimization – Reducing Memory Usage Per Application

- Having a software backend helps (GL consumes more than 10MB)
- Letting the kernel choose when to throw away data helps (mmap)
- Reducing memory duplication per process is a first step
  - String share
  - Eet
  - Image/Font cache
  - Copy On Write (When introduced saved 10% of memory and gain 5% on CPU)

Optimization - Reducing Memory Usage Globally is Necessary

- Every application using EFL has at least one thing in common

  EFL!

- Meaning that part of the libraries' initialization could be shared

  Welcome quicklaunch!

- Applications that use the same system theme will be similar visually
- Image/Glyph/Font geometry are costly to load and consume memory
- Sharing is doable, but tricky

  Welcome Cserve2!

Optimization - Quicklaunch

Pros:

- Libraries are preloaded in memory
- Link is done once
- Partial initialization done once
- Preload library in memory at boot time

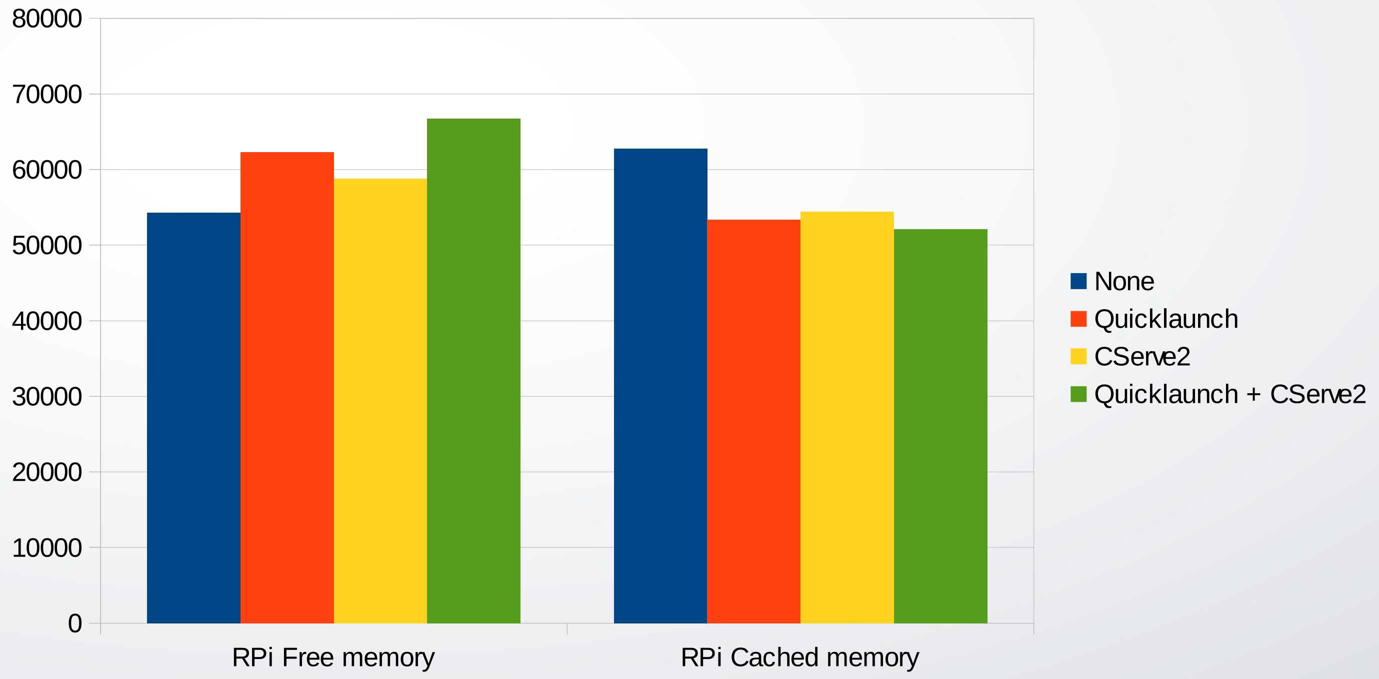
Cons:

- Tools need to read the updated argv[0]
- Quicklaunch does some of the job of systemd –user

Optimization - Cserve2

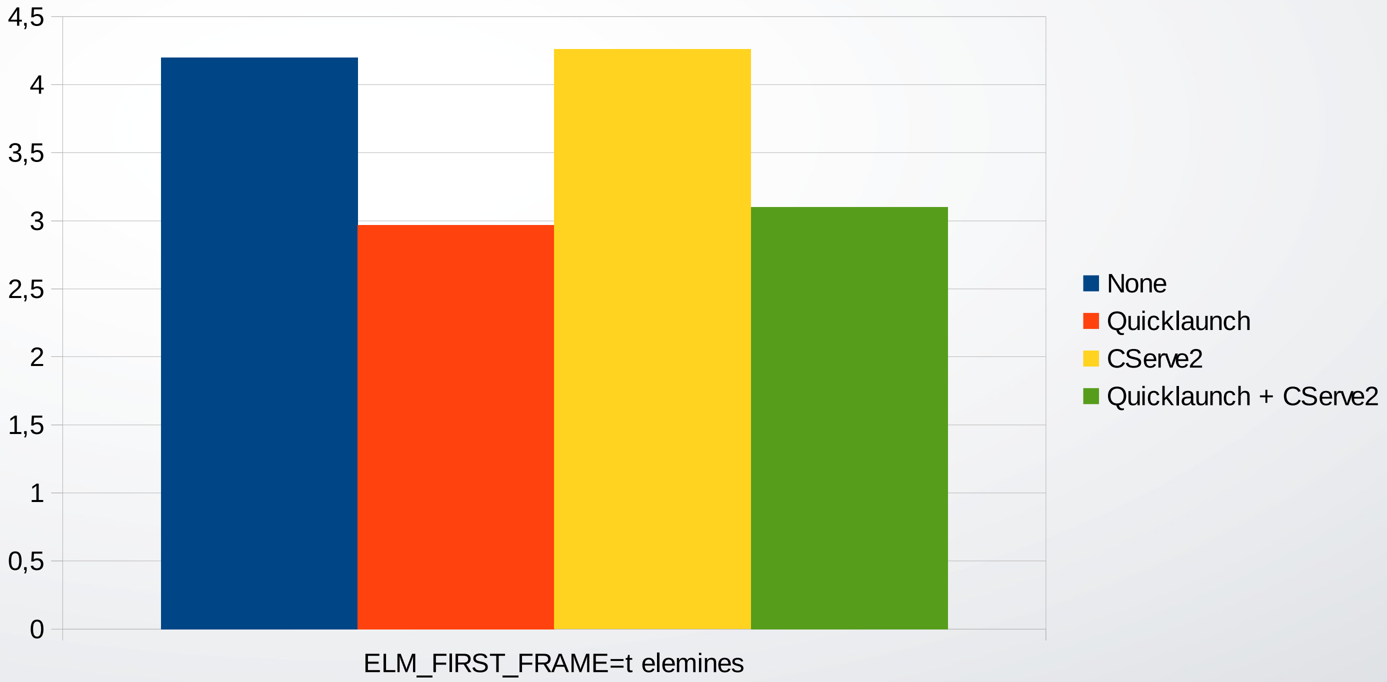
- Pros:
  - Decode image/glyph once
  - Share pixel data
  - Faster startup time, reduced memory usage
- Cons:
  - Security credential progapagtion isn't implemented, only user level
  - No integration with kernel to act in OOM situation
  - Difficult to do restart on crash without crashing client
  - Requires client to have the same theme!!!

Optimization – Impact of Preloading and Shared Cache Infrastructure



TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

Optimization – Impact of Preloading and Shared Cache Infrastructure



Optimization – Power Consumption on RPi

- Normal = 2W * 4.20s
- Quicklaunch = 2W * 2.97s
- Cserve2 = 2W * 4.26s
- Quicklaunch + Cserve2 = 2W * 3.10s

  - (Energy consumption is an average)

Optimization – Quicklaunch

- Preloading and partial initialization of EFL does pay off
- Could be improved by preconnecting to X/Wayland and preloading the theme
- Need to find a proper fix to integrate with systemd user session

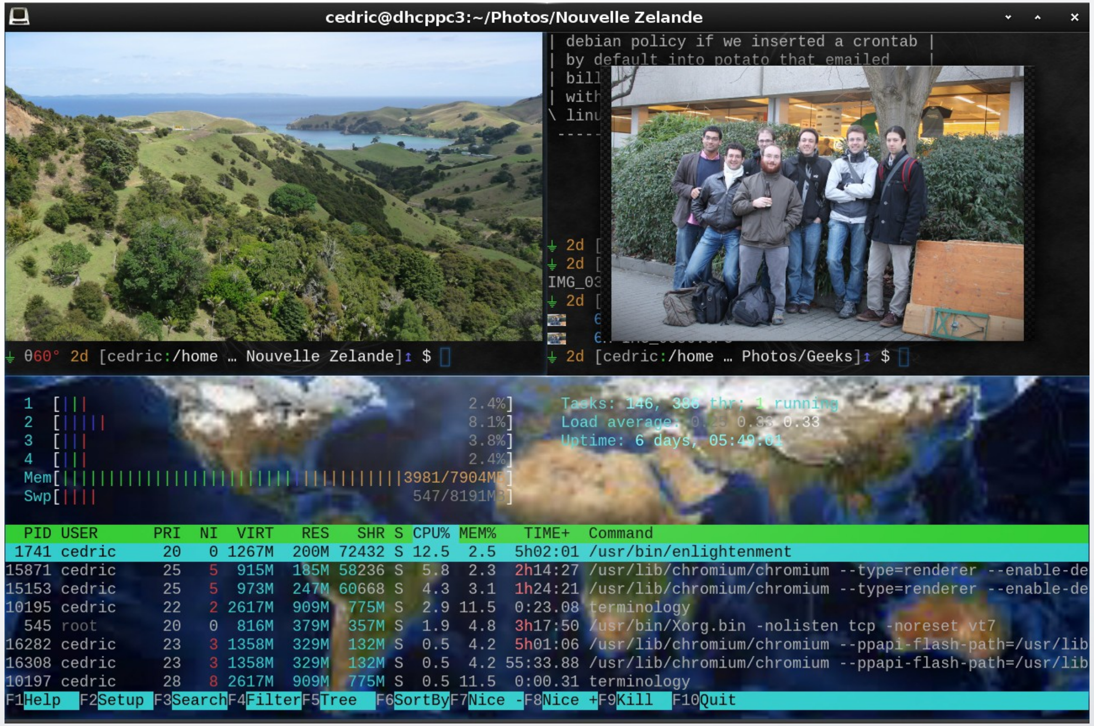
Optimization - Cserve2

- High upfront cost
- Does not share texture (possible with Wayland)
- Still a lot of potential, but requires work:
  - Cache of disk RAW image when applications are minized
  - Detect usual images needed by applications and load them as soon as the applications start

Optimization – Single Instance

- Alternative to preloading: single instance
- Use case, one process, multiple window: *Terminology*
- Doesn't apply to all applications
- Doesn't provide process separation…
- One goes down, everyone goes down !

Terminology

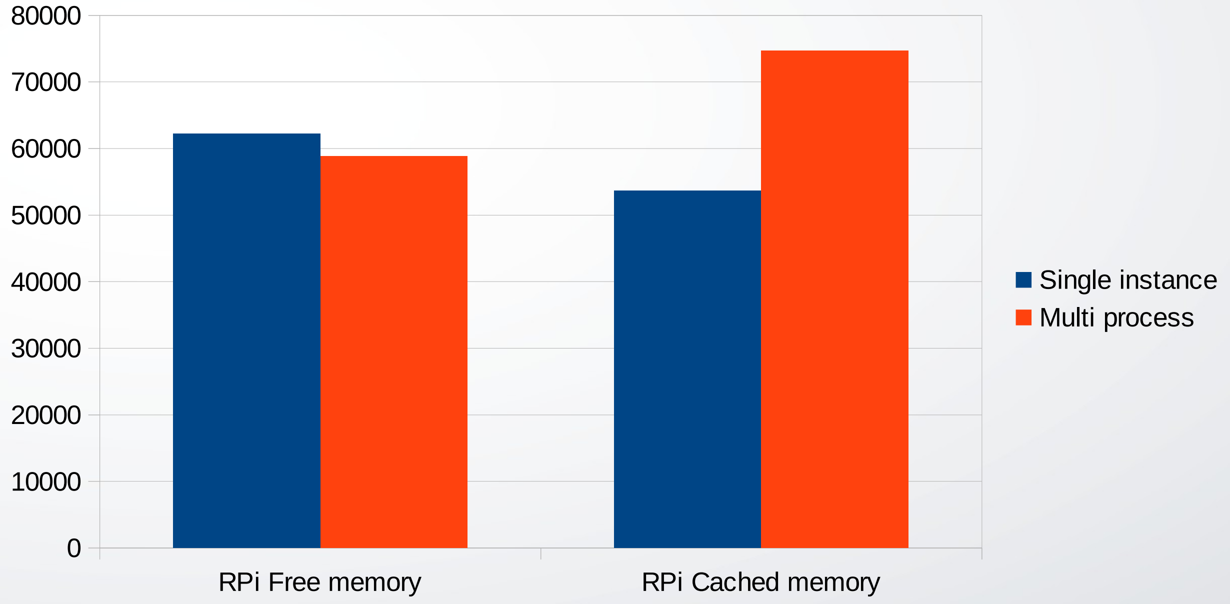

cedric@dhcppc3:~/Photos/Nouvelle Zelande

```
| debian policy if we inserted a crontab |
| by default into potato that emailed    |
| bill
| with
\ linu
 -----

⏚ 2d [
⏚ 2d [
IMG_03
⏚ 2d [
       6
       6.
⏚ 2d [cedric:/home … Photos/Geeks]↥ $
```

```
⏚ 060° 2d [cedric:/home … Nouvelle Zelande]↥ $
```

```
  1  [|||                                 2.4%]   Tasks: 146, 386 thr; 1 running
  2  [|||||                               8.1%]   Load average: 0.25 0.33 0.33
  3  [|||                                 3.8%]   Uptime: 6 days, 05:49:01
  4  [|||                                 2.4%]
  Mem[||||||||||||||||||||||||||||||||||3981/7904MB]
  Swp[||||                            547/8191MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 1741 cedric     20   0 1267M  200M 72432 S 12.5  2.5  5h02:01 /usr/bin/enlightenment
15871 cedric     25   5  915M  185M 58236 S  5.8  2.3  2h14:27 /usr/lib/chromium/chromium --type=renderer --enable-de
15153 cedric     25   5  973M  247M 60668 S  4.3  3.1  1h24:21 /usr/lib/chromium/chromium --type=renderer --enable-de
10195 cedric     22   2 2617M  909M  775M S  2.9 11.5  0:23.08 terminology
  545 root       20   0  816M  379M  357M S  1.9  4.8  3h17:50 /usr/bin/Xorg.bin -nolisten tcp -noreset vt7
16282 cedric     23   3 1358M  329M  132M S  0.5  4.2  5h01:06 /usr/lib/chromium/chromium --ppapi-flash-path=/usr/lib
16308 cedric     23   3 1358M  329M  132M S  0.5  4.2 55:33.88 /usr/lib/chromium/chromium --ppapi-flash-path=/usr/lib
10197 cedric     28   8 2617M  909M  775M S  0.5 11.5  0:00.31 terminology
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Optimization – Single Instance vs Multi Process



Optimization – Single Instance

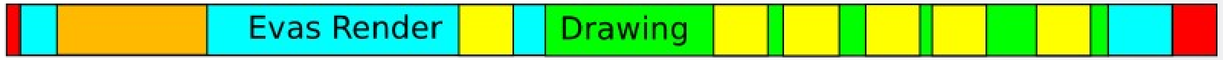
- It improves cost, especially for big desktop applications
- Potential for systemd –user session
- Has limitation that can't be overcome, so not a generic solution

- EFL community cares about performance
- Samsung cares about performance for all product lines
- There is always room for improvement
- Evas scenegraph logic is more than 10 years old, refactoring and cleaning is necessary
- Improving our tests suites and especially our benchmark is a permanent task
- Never ending story! Always something to improve!

Evas, Let's Rewrite Everything!

- Built at a time where SMP wasn't common
- Linux Kernel scheduler is improving
- Learned a lot about what we can do with the hardware we have to reduce memory, cpu and battery consumption!

Evas – Current Rendering Pipeline



🟨 computing CPU intensive data

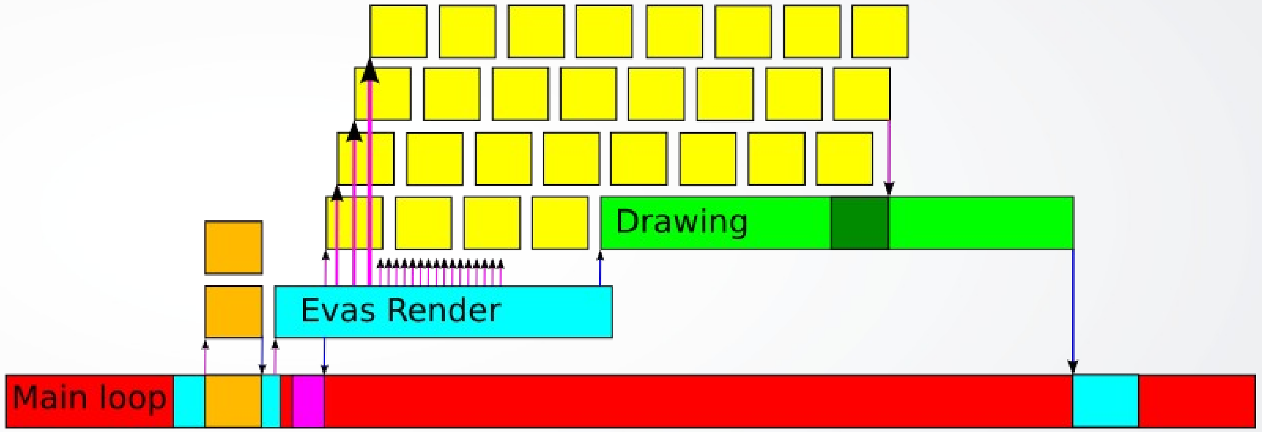
🟩 compositing data, mostly memory bound

🟧 layout object

🟦 walk tree to order operation

🟥 application logic

Evas – New Pipeline Design



🟨 computing CPU intensive data

🟩 compositing data, mostly memory bound

🟧 layout object

🟦 walk tree to order operation

🟥 application logic

🟪 waiting for COW

Possible Areas of Improvement

- Integration of Cserve2 with Wayland
- Generaly improve Cserve2 (RAW image cache, application image preloading, etc.)
- Add relative positionning support
- Add hardware layer support
- Provide a way to cache flattened structure across applications (useful for theme, icons description, strings, ...)

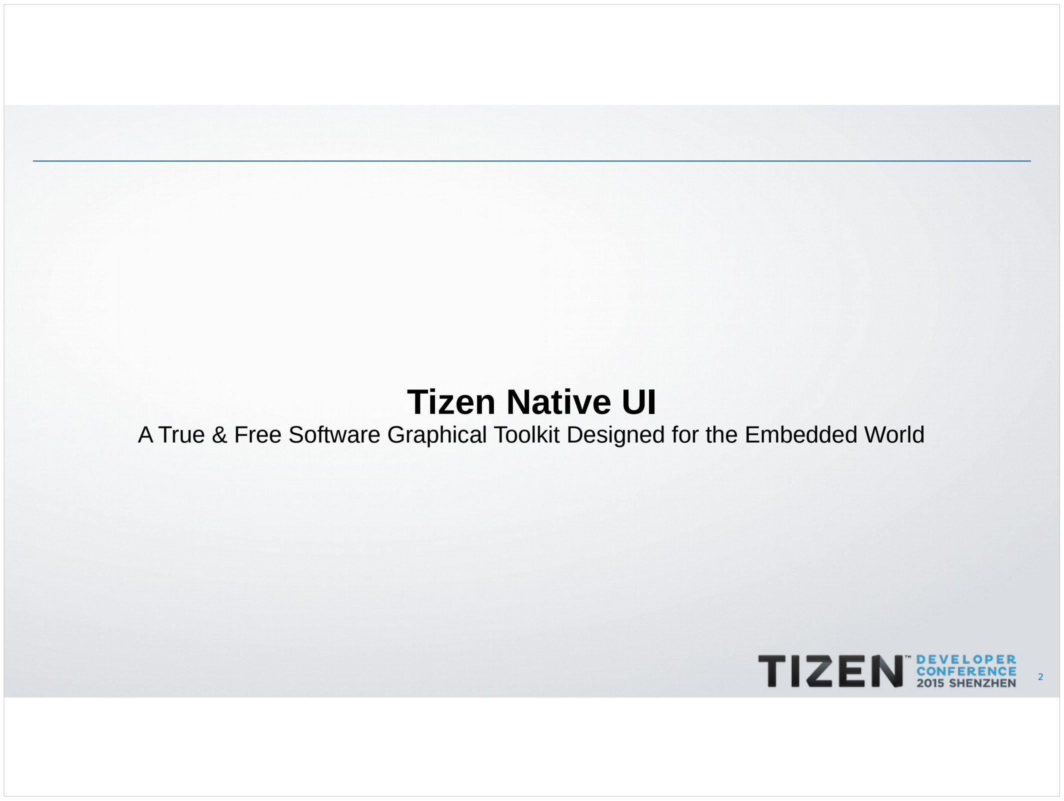
There's Still a Lot of Work to Do!

- Contributions are welcome from all around the world!
- True, open source project with a community; everyone can c ontribute directly!
- More companies are using it other than Samsung
- Join the mailing list and IRC!
- And poke me around any time during the event if you have question!

Questions?

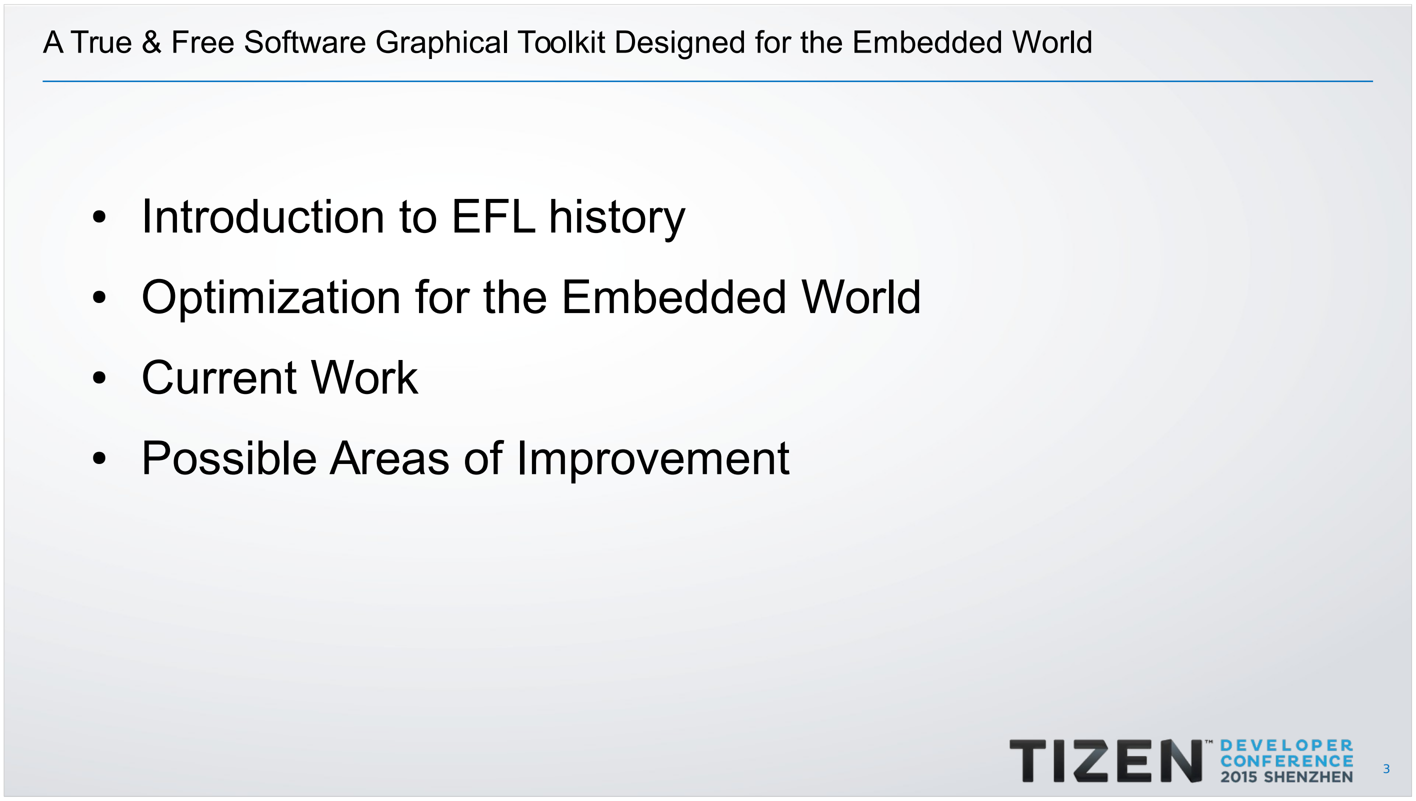
Cedric BAIL

Samsung Research America



Tizen Native UI

A True & Free Software Graphical Toolkit Designed for the Embedded World

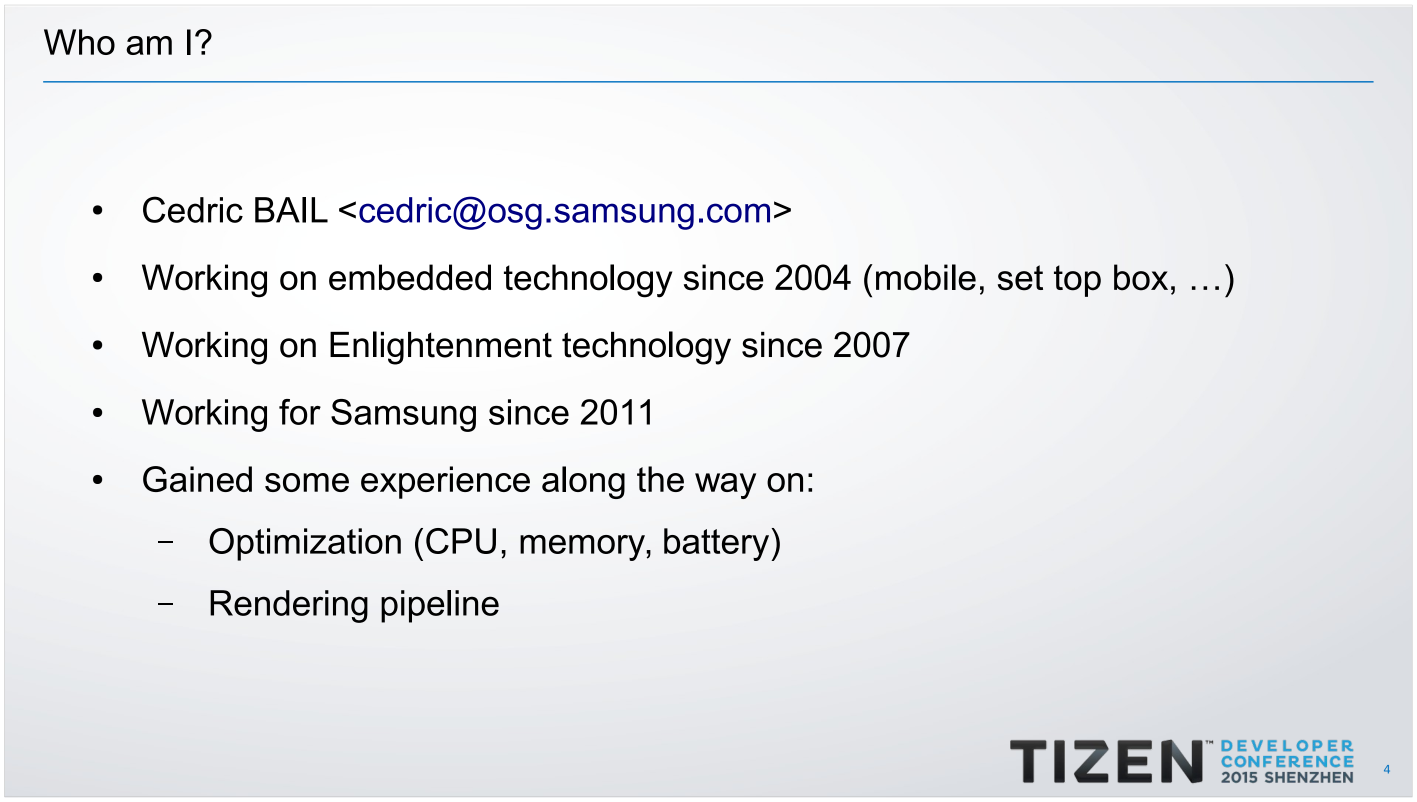


A True & Free Software Graphical Toolkit Designed for the Embedded World

- Introduction to EFL history
- Optimization for the Embedded World
- Current Work
- Possible Areas of Improvement

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

The plan of this talk is going to be



Who am I?

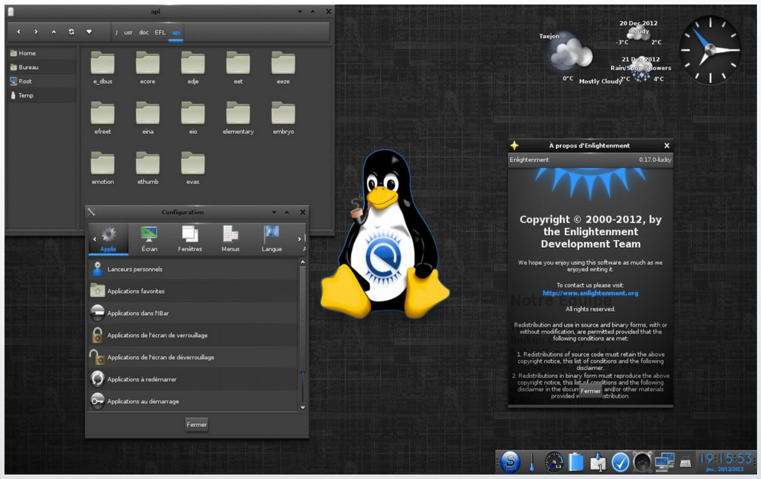
- Cedric BAIL <cedric@osg.samsung.com>
- Working on embedded technology since 2004 (mobile, set top box, …)
- Working on Enlightenment technology since 2007
- Working for Samsung since 2011
- Gained some experience along the way on:
  - Optimization (CPU, memory, battery)
  - Rendering pipeline

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

I am working for Samsung Open Source Group on EFL
I have been working on embedded technology for more than a decade now

Tizen Native UI – EFL, Enlightenment Foundation Libraries

- What are the Enlightenment Foundation Libraries?
  - Toolkit created for Enlightenment 17

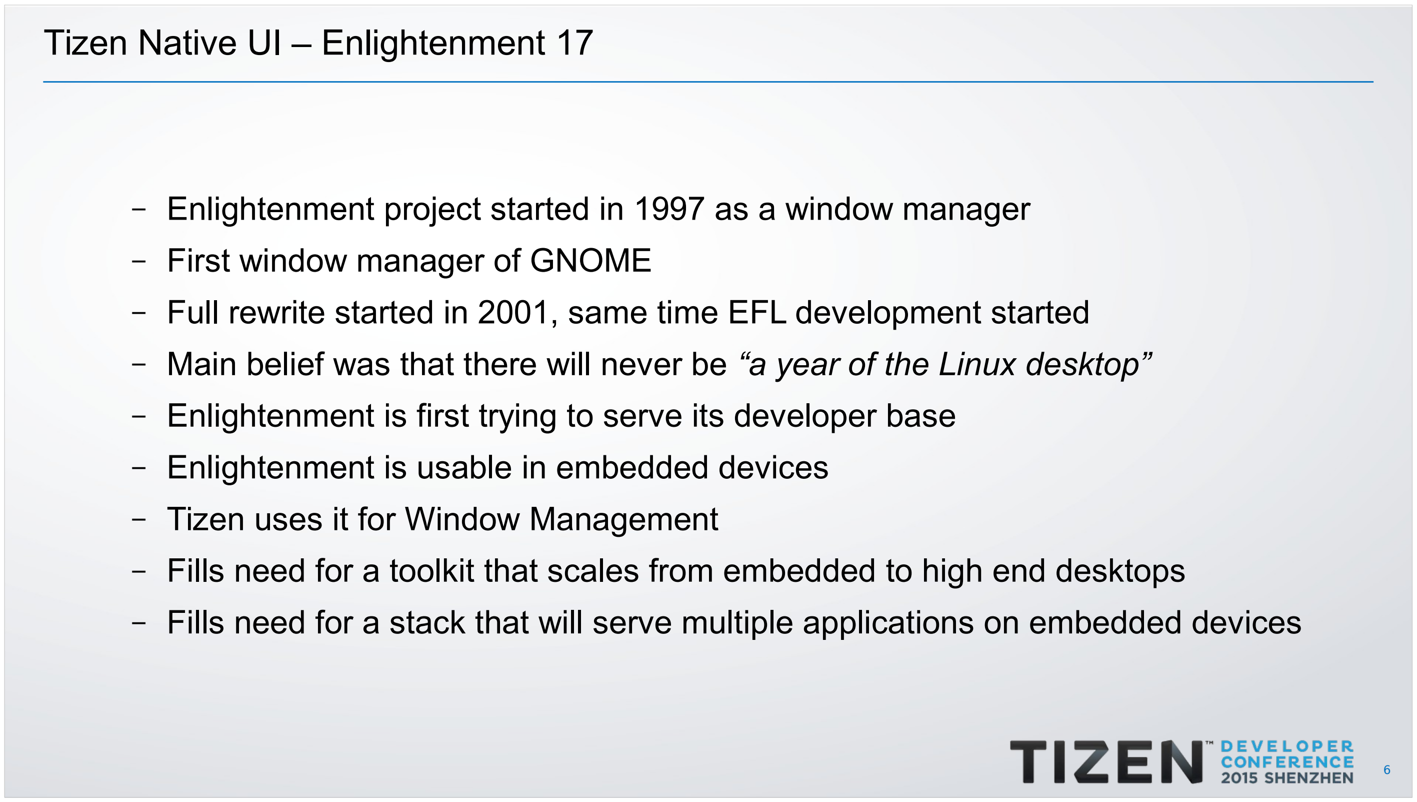


TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

So what is this EFL, Enlightenment Foundation Libraries ? Where did it comes from ?

It is a toolkit, a set of libraries, that was created over the last decade to develop Enlightenment 17

Now you are asking, what is Enlightenment 17 ! Right, so in the Linux world we use a X server to display stuff on screen this day. This X server doesn't have any policy regarding window placement, window border and background. This is the job of the Window Manager, and that what Enlightenment does.



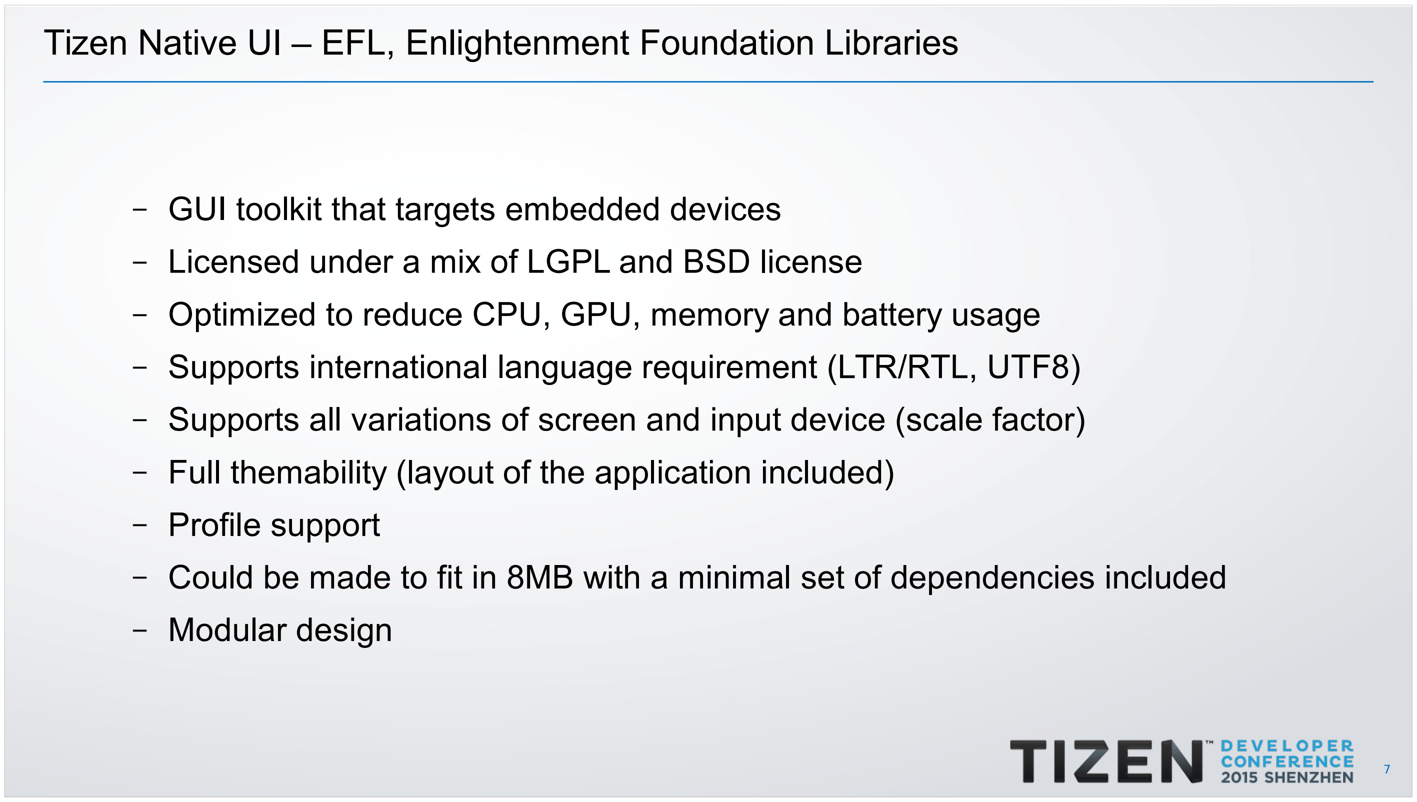
Tizen Native UI – Enlightenment 17

- Enlightenment project started in 1997 as a window manager
- First window manager of GNOME
- Full rewrite started in 2001, same time EFL development started
- Main belief was that there will never be *“a year of the Linux desktop”*
- Enlightenment is first trying to serve its developer base
- Enlightenment is usable in embedded devices
- Tizen uses it for Window Management
- Fills need for a toolkit that scales from embedded to high end desktops
- Fills need for a stack that will serve multiple applications on embedded devices

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

Now that we know a little bit what Enlightenment 17 is, a little bit more background about it and its community

It is an open source/free software



Tizen Native UI – EFL, Enlightenment Foundation Libraries

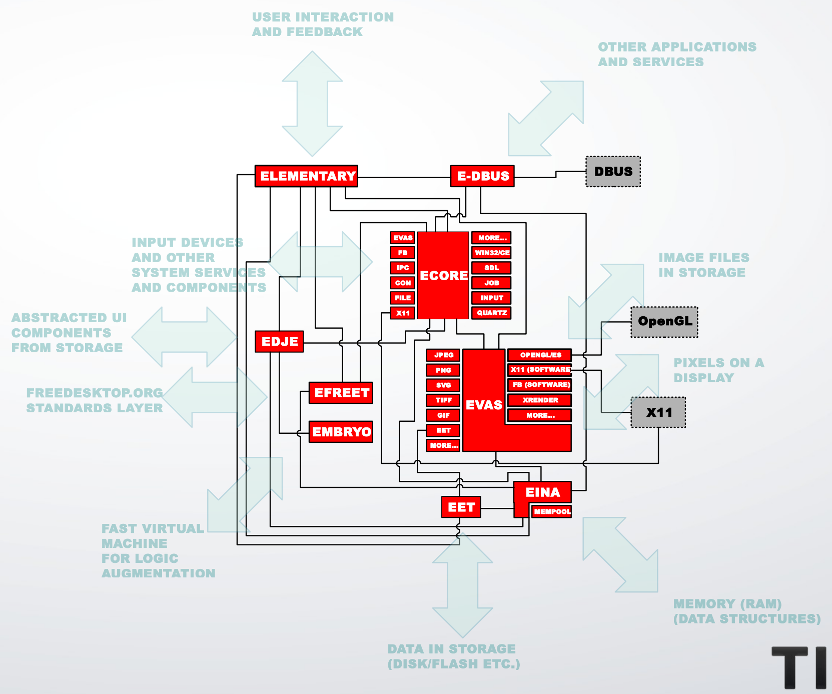
- GUI toolkit that targets embedded devices
- Licensed under a mix of LGPL and BSD license
- Optimized to reduce CPU, GPU, memory and battery usage
- Supports international language requirement (LTR/RTL, UTF8)
- Supports all variations of screen and input device (scale factor)
- Full themability (layout of the application included)
- Profile support
- Could be made to fit in 8MB with a minimal set of dependencies included
- Modular design

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 7

But for that goal, we needed a new toolkit. Nothing when we started EFL existed that could meet our need and there was no project that could be moved to be aligned to our goal as a community.

So we did spend a decade writing a new toolkit that scale from the heaviest desktop to the smallest embedded device and give you the best your hardware can do.

Tizen Native UI – EFL, Enlightenment Foundation Libraries


USER INTERACTION AND FEEDBACK
OTHER APPLICATIONS AND SERVICES
DBUS
ELEMENTARY
E-DBUS
INPUT DEVICES AND OTHER SYSTEM SERVICES AND COMPONENTS
EVAS
FB
IPC
CON
FILE
X11
ECORE
MORE...
WIN32/CE
SDL
JOB
INPUT
QUARTZ
IMAGE FILES IN STORAGE
OpenGL
ABSTRACTED UI COMPONENTS FROM STORAGE
EDJE
JPEG
PNG
SVG
TIFF
GIF
EET
MORE...
EVAS
OPENGL/ES
X11 (SOFTWARE)
FB (SOFTWARE)
XRENDER
MORE...
PIXELS ON A DISPLAY
X11
FREEDESKTOP.ORG STANDARDS LAYER
EFREET
EMBRYO
EINA
MEMPOOL
EET
FAST VIRTUAL MACHINE FOR LOGIC AUGMENTATION
MEMORY (RAM) (DATA STRUCTURES)
DATA IN STORAGE (DISK/FLASH ETC.)

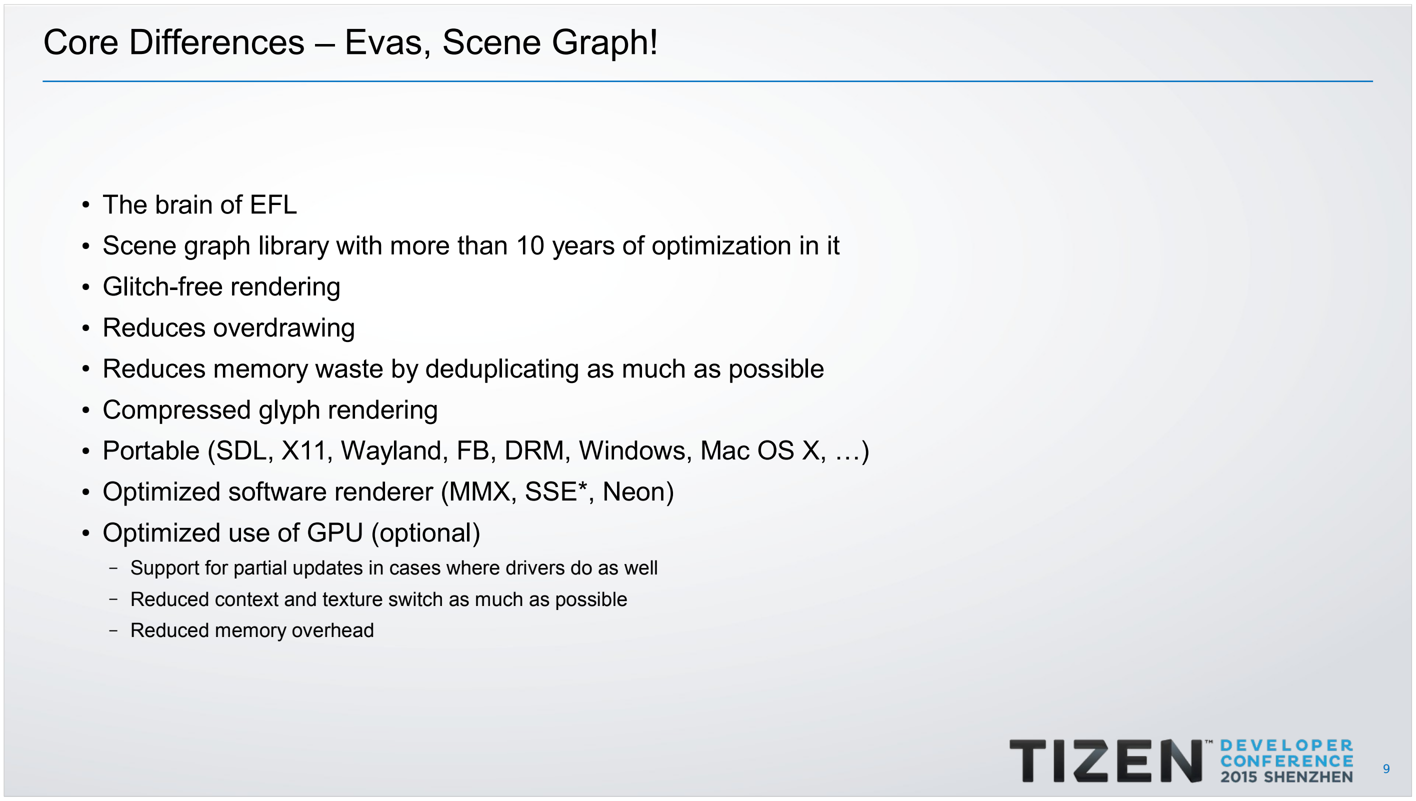
TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 8

I always prefer to show this diagram of how EFL component interact than the classic layer way of displaying any stack of software. This make it clear that has a developer you have access to every component of that stack directly and that they interconnect each other in many way.

The main 3 components and almost the reason of existence of EFL, is Evas, EFL scenegraph library, Edje, the theme library, and Elementary, the widget library. I will be more specific later on this component.

The other component that you can see, that are actually more classic :
- Eina, data type library
- Ecore, event library
- Efreet, freedesktop library
- Embryo, scripting library
- Eldbus, dbus library
- Eet, serialization libray

There are a few more that are not on this graph, I will let you discover them. We do have some doc now :-)



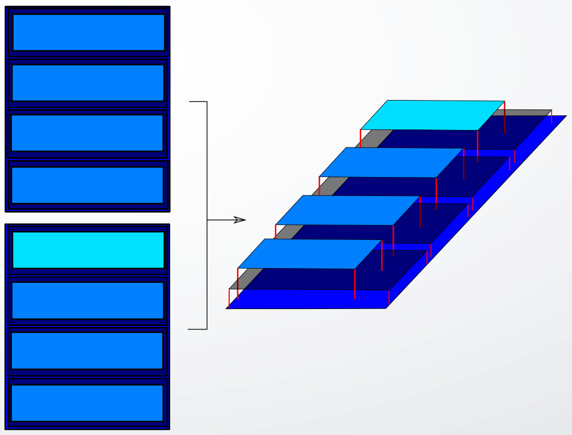
Core Differences – Evas, Scene Graph!

- The brain of EFL
- Scene graph library with more than 10 years of optimization in it
- Glitch-free rendering
- Reduces overdrawing
- Reduces memory waste by deduplicating as much as possible
- Compressed glyph rendering
- Portable (SDL, X11, Wayland, FB, DRM, Windows, Mac OS X, …)
- Optimized software renderer (MMX, SSE*, Neon)
- Optimized use of GPU (optional)
  - Support for partial updates in cases where drivers do as well
  - Reduced context and texture switch as much as possible
  - Reduced memory overhead

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

So Evas, this is the main difference to any other toolkit. The « raison d'etre » of EFL.

Core Differences – Evas, Scene Graph!



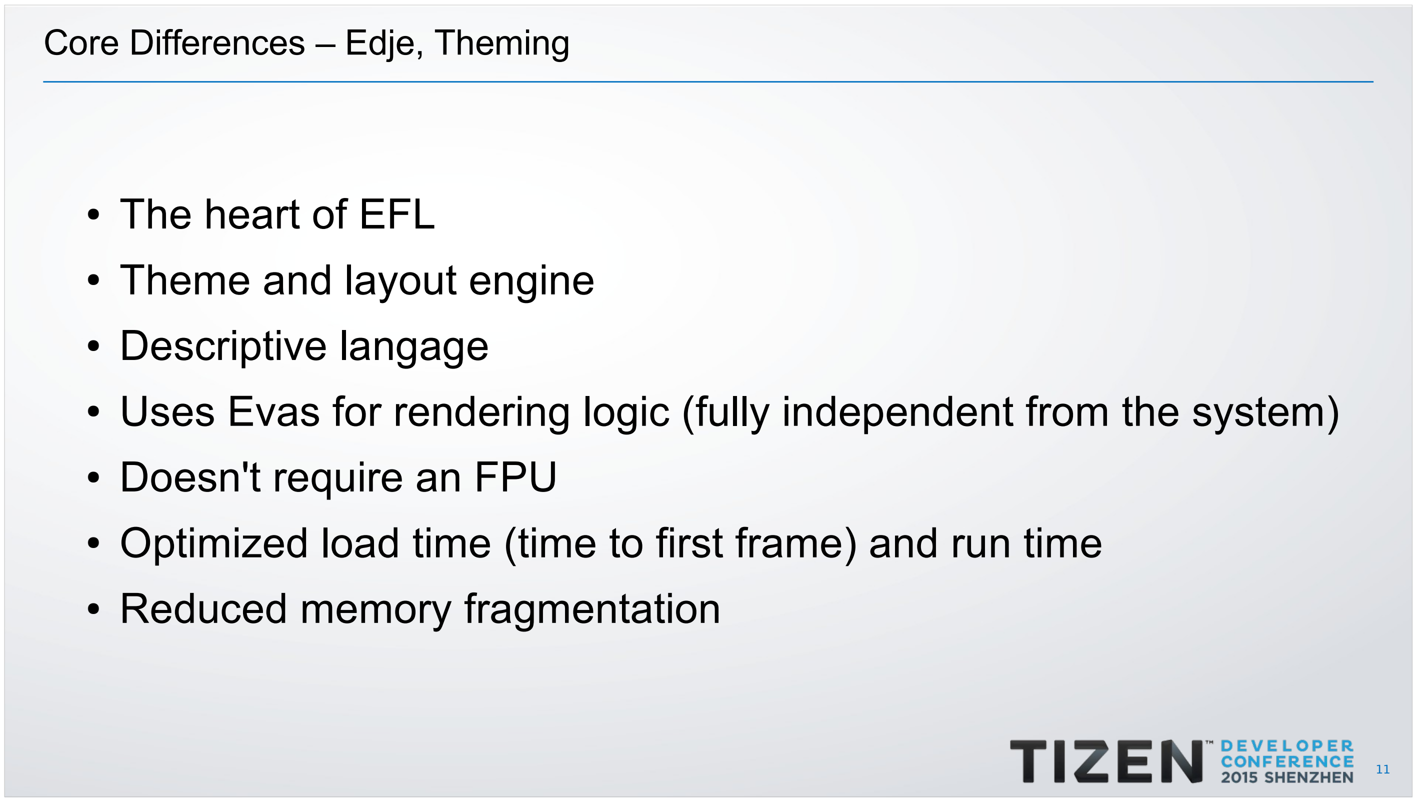
But what is a Scene Graph ? Where does that come from ?

It is something most game engine use with various trick in. I will cover only here how Evas implement its Scene Graph, but there is a huge amount of litterature on the subject if you are interested.

So Evas does keep a graph of the previous set of object and will compare that to the one that need to be rendered on screen for this frame. This give it the possibility to do nice trick.

Like shown here, where you will only update what change between two frame. Having a graph make it possible to also only draw the area that are really needed. We can also reorder the comand we send to the GPU to limit the amount of Texture and Shader change.

This structure enable a lot of optimization possibility that I don't have time to cover here, but keep in mind that we do this by keeping our memory foot print low by using various deduplication technique. Like refcounting data and copy on write.



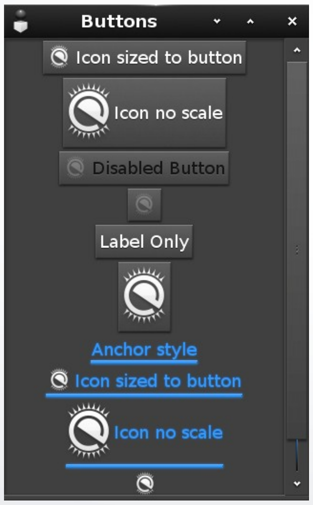
Core Differences – Edje, Theming

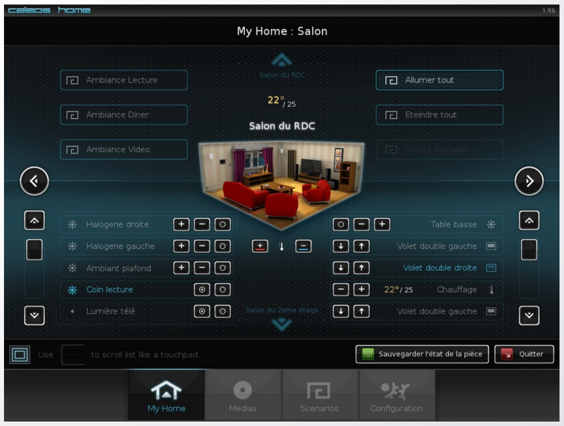
- The heart of EFL
- Theme and layout engine
- Descriptive langage
- Uses Evas for rendering logic (fully independent from the system)
- Doesn't require an FPU
- Optimized load time (time to first frame) and run time
- Reduced memory fragmentation

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 11

Very specific to EFL
Predate a lot of other solution and did inspire other (Like Qt/QML solution)

Core Differences – Edje, Theming



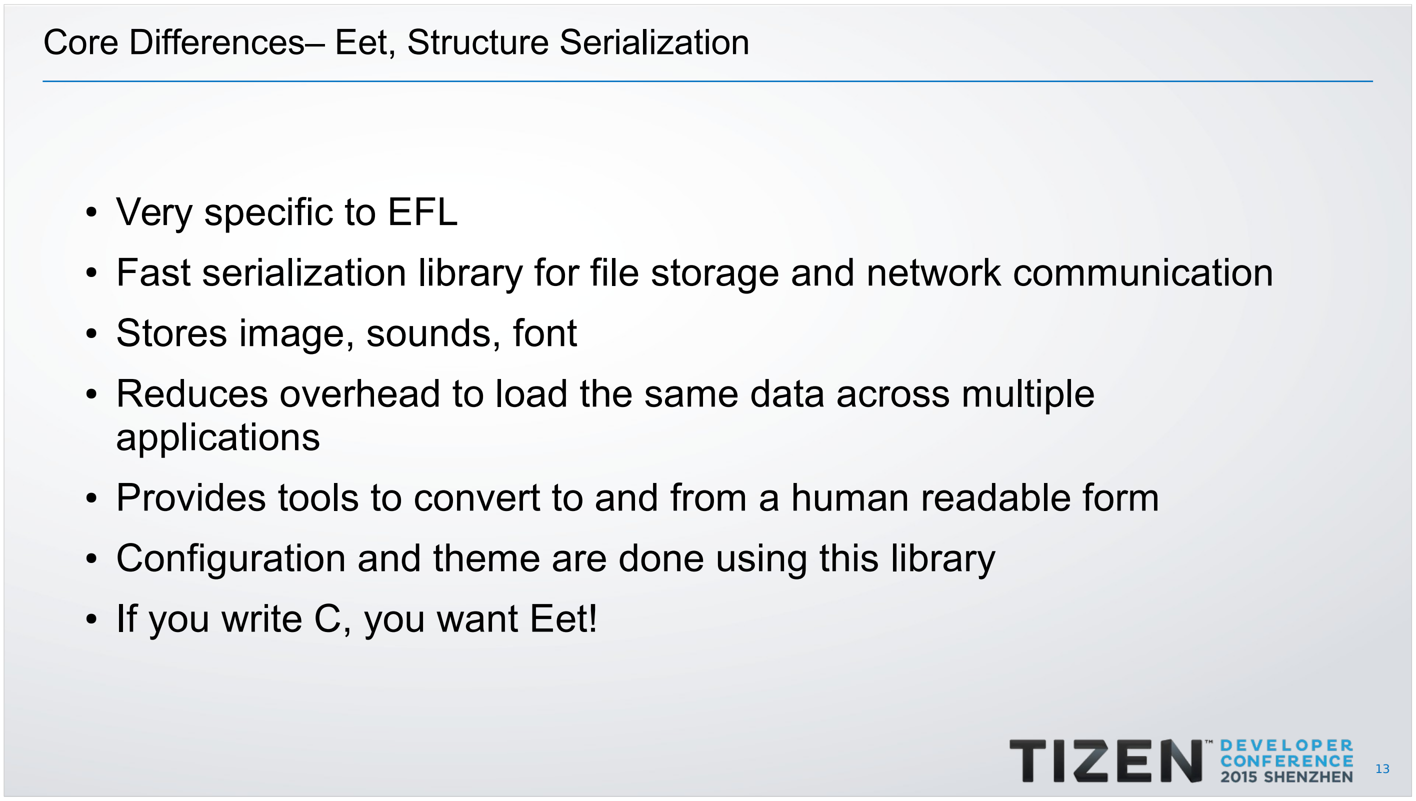


TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

As you can see here is multiple example of the same button using different theme. Some even have different effect, like mouse over effect, that are fully implemented in the theme.

On the left is elementary default theme used in elementary test application which display every elementary widget

On the right is calaos, a home automation solution which is using elementary widget for its user interface without looking at all like a desktop software.



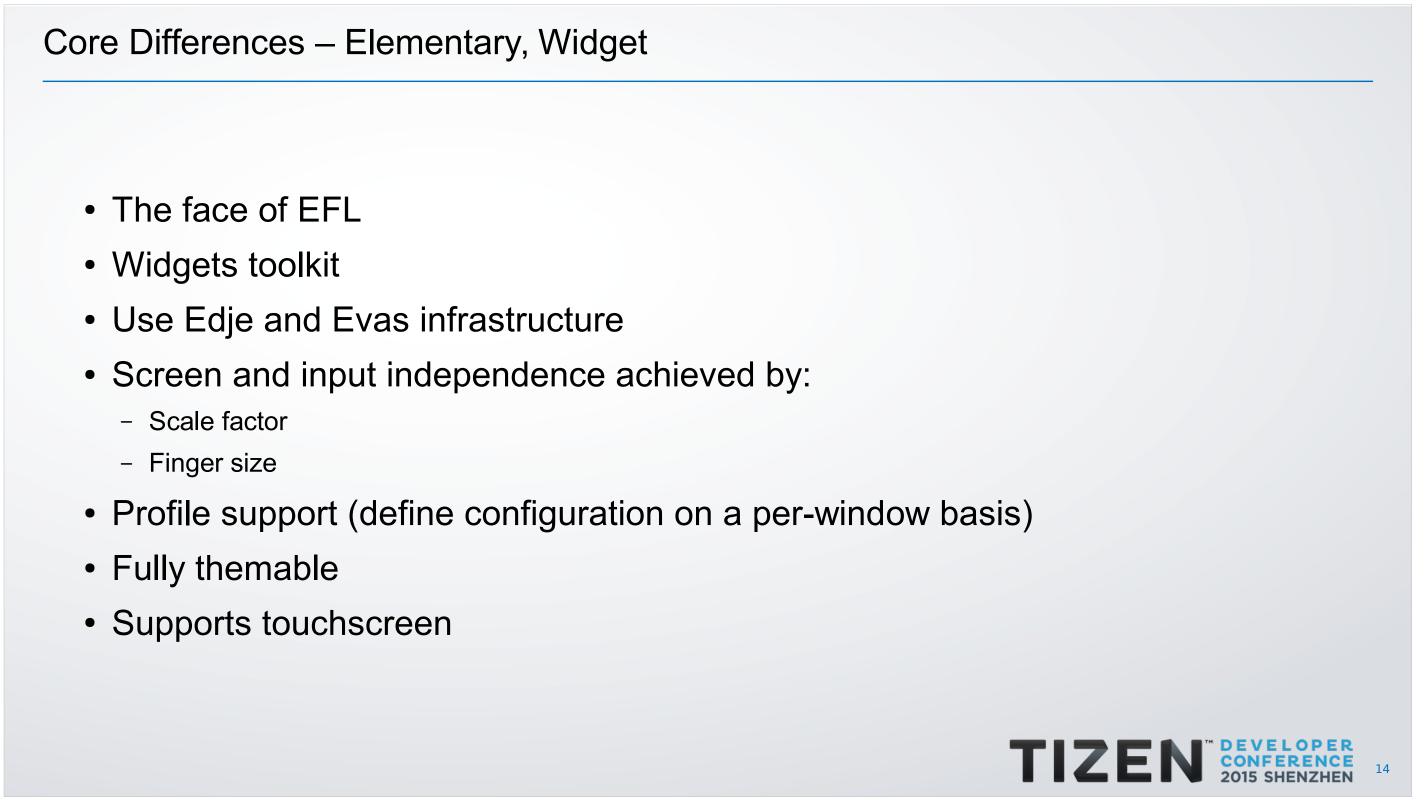
Core Differences– Eet, Structure Serialization

- Very specific to EFL
- Fast serialization library for file storage and network communication
- Stores image, sounds, font
- Reduces overhead to load the same data across multiple applications
- Provides tools to convert to and from a human readable form
- Configuration and theme are done using this library
- If you write C, you want Eet!

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 13

Oh, I forgot to tell you about this little « trick » library.

You basically just give it a pointer to a C structure and presto it is serialized on the disk, network or in memory. Of course the reverse work just as well, you request to deserialize some data from disk, network or memory and presto, you just got a pointer to your C structure. Pretty handy if you do C development !



Core Differences – Elementary, Widget

- The face of EFL
- Widgets toolkit
- Use Edje and Evas infrastructure
- Screen and input independence achieved by:
  - Scale factor
  - Finger size
- Profile support (define configuration on a per-window basis)
- Fully themable
- Supports touchscreen

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 14

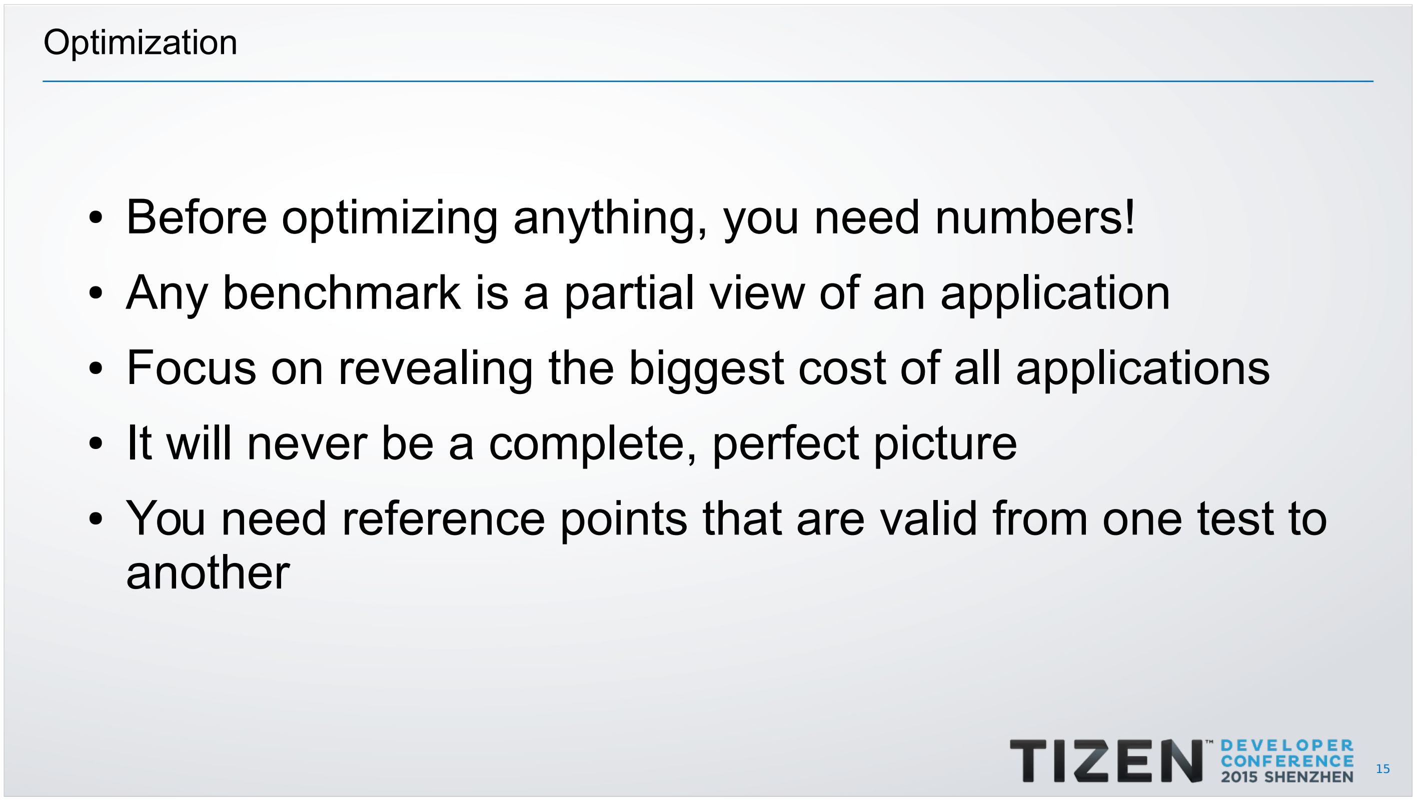
Actually that's the fourth widget set we write using EFL. That's what happen when writting a widget got to easy !

This iteration is quite close to what we really had in mind, something that can be used from the lower end device to the most beasty desktop computer. It is a fully scalable widget set that should make any kind of application possible.

There is two things that are important when we mean scale. First it needs to scale its ressource usage to fit on low end device, but also on high end by using all the possible hardware component to be as efficient per watt as possible.

Second it needs to « scale » the UI and adapt to the device context. How/where it is used. What the resolution of the device. What the input method, its resolution, …

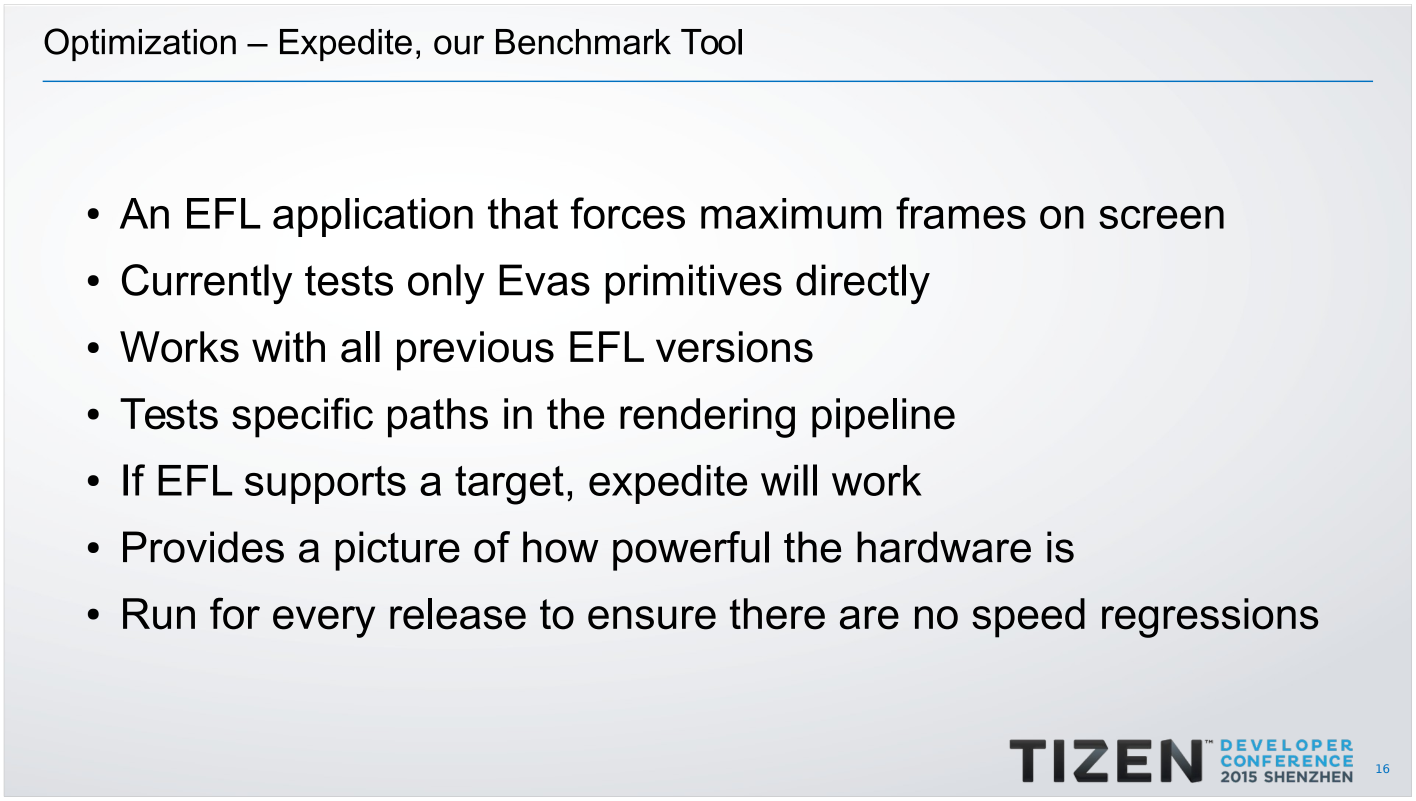
And all of that need to be dynamic !


Optimization

- Before optimizing anything, you need numbers!
- Any benchmark is a partial view of an application
- Focus on revealing the biggest cost of all applications
- It will never be a complete, perfect picture
- You need reference points that are valid from one test to another

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

So back to the topic now that you have some background :-) I mean, I have been telling you that EFL is heavily optimized, but what does optimization mean at all !



Optimization – Expedite, our Benchmark Tool

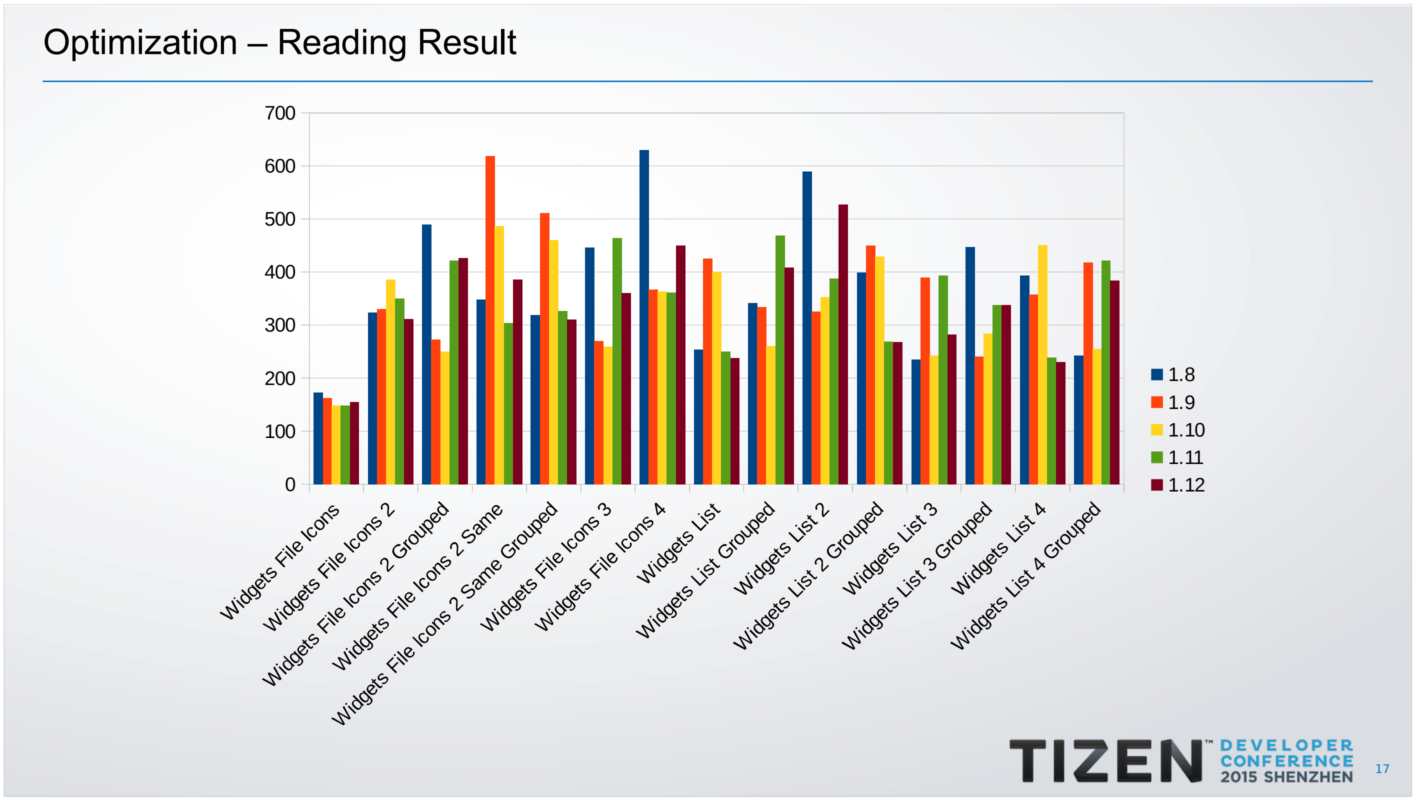
- An EFL application that forces maximum frames on screen
- Currently tests only Evas primitives directly
- Works with all previous EFL versions
- Tests specific paths in the rendering pipeline
- If EFL supports a target, expedite will work
- Provides a picture of how powerful the hardware is
- Run for every release to ensure there are no speed regressions

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

We have a simple tool designed to micro benchmark some part of the rendering pipeline : « expedite »

It doesn't test Edje or Elementary (It could, just never had the time to add test for)

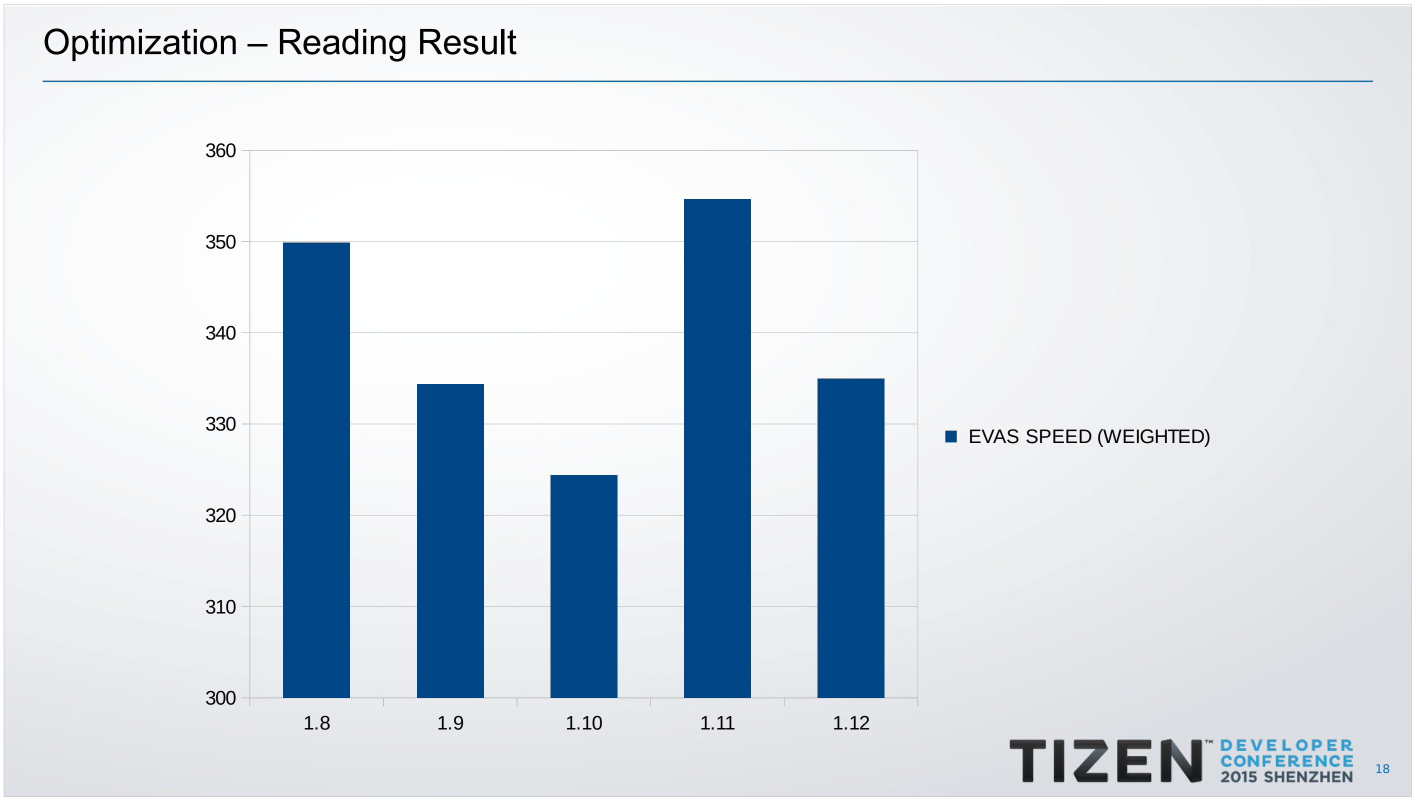
Test is limited ! It only test what it test ! But it is going to give you a good picture of what your hardware can do as this is the first application that is going to run once you have EFL setup on a system.



Optimization – Reading Result

So what kind of result do we get… pretty random, no ?

Things go up and down for every release on every test. We do not get a constant result, mostly due to very little variation on the CPU, kernel, memory and the whole system state ! Basically any variation that is < 5 % is the result of some glitch somewhere in the system that you can't really know why.

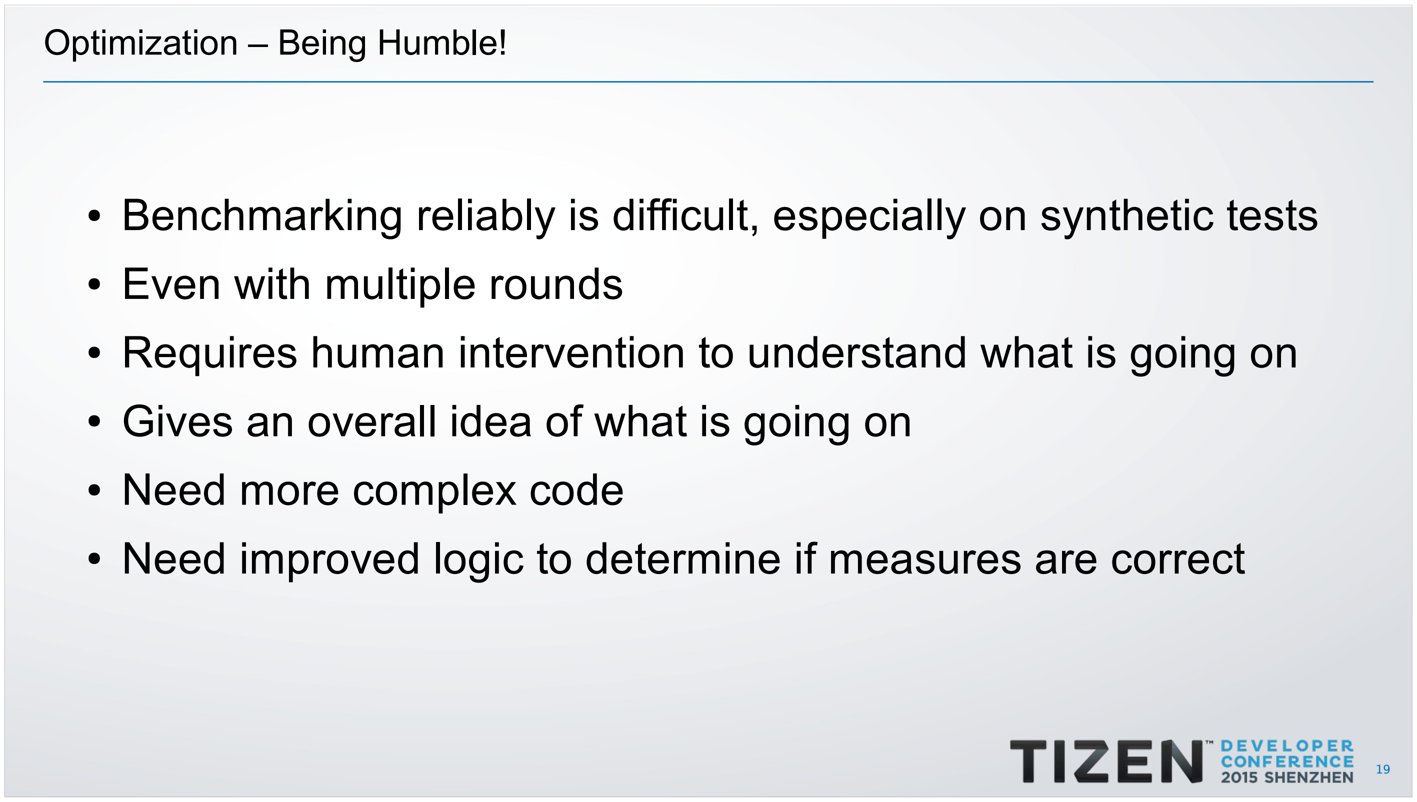


Optimization – Reading Result

So we actually look at the aggregated score over time. Here is all last year release and even by adjusting for this 5 % error margin, 1.10 was actually quite slower compared to 1.8 (from 350 ± 5% to 325 ± 5%). So in 1.11 I did track where that slow down was coming from and boom we were back on track in term of performance.

This is a never ending game. We have to constantly test and benchmark efl to see what is going on and fix speed regression as we see it…

But as I said, if you don't measure it, you are not going to fix a problem. This is always partial and we have to improve what we measure to detect issue that does matter.



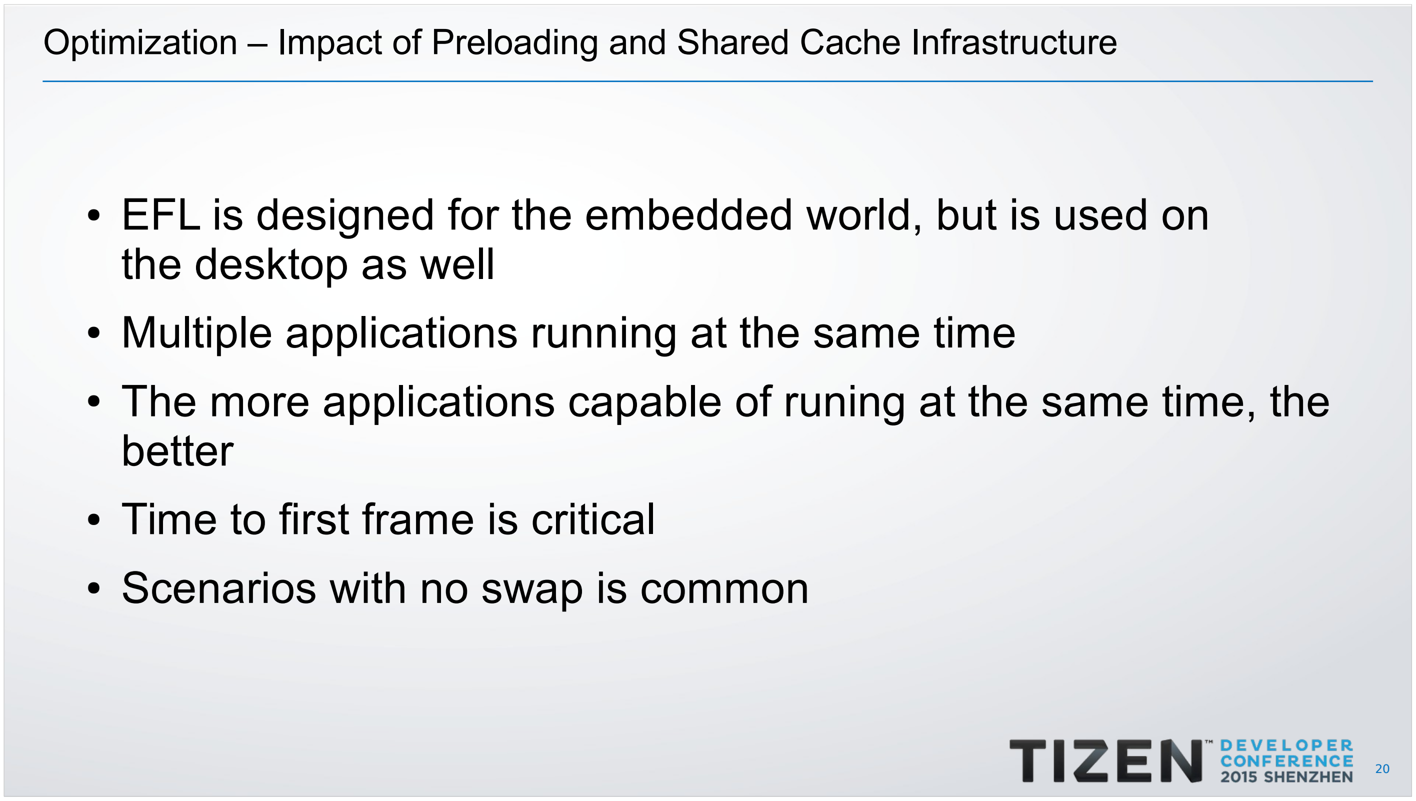
Optimization – Being Humble!

- Benchmarking reliably is difficult, especially on synthetic tests
- Even with multiple rounds
- Requires human intervention to understand what is going on
- Gives an overall idea of what is going on
- Need more complex code
- Need improved logic to determine if measures are correct

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

As you already noticed, benchmarking is tricky !

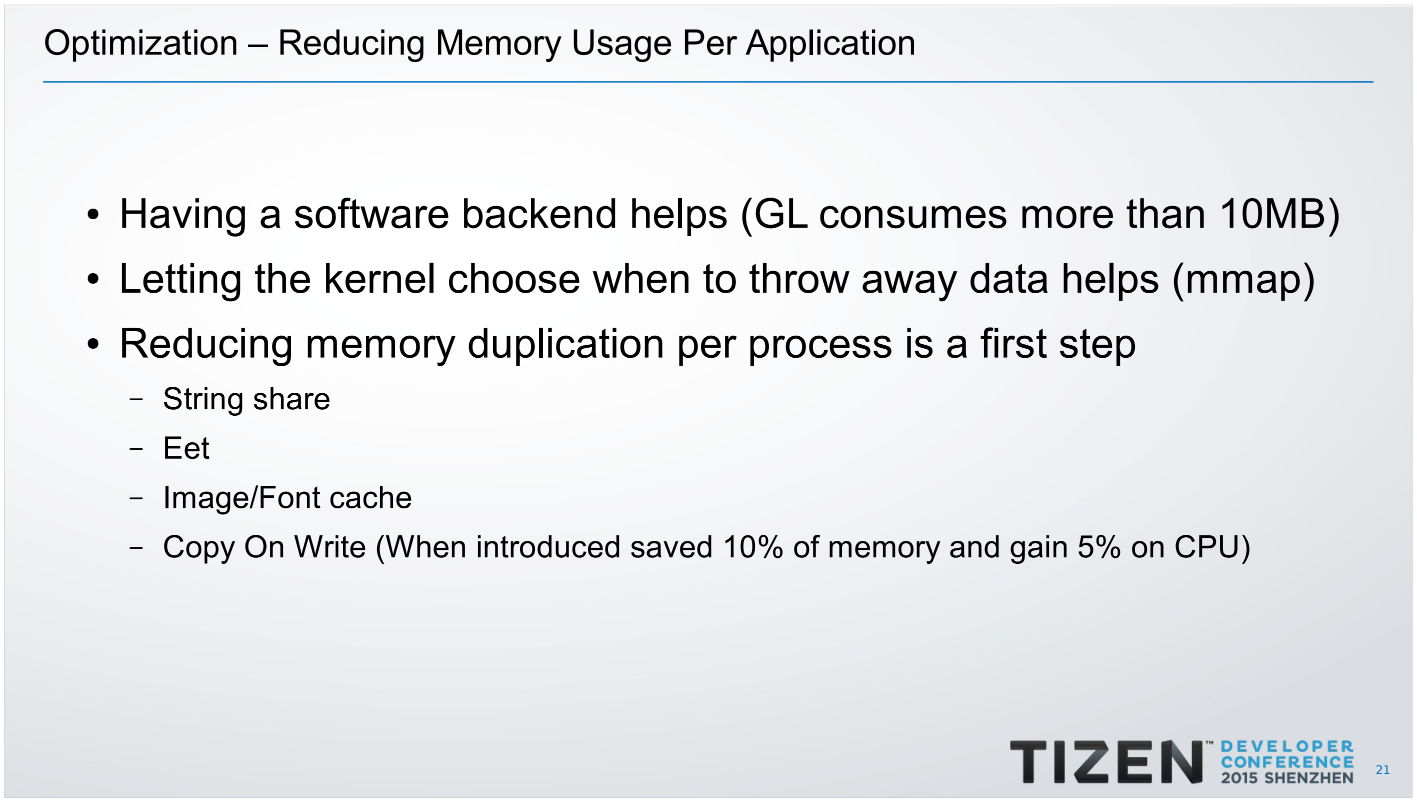
Synthetic tests tend to trigger a lot of subsystem, even kernel scheduler has a huge impact on the result ! Meaning, if you compare something, make sure that everything is identical, including the software stack up to what you are testing !



Optimization – Impact of Preloading and Shared Cache Infrastructure

- EFL is designed for the embedded world, but is used on the desktop as well
- Multiple applications running at the same time
- The more applications capable of runing at the same time, the better
- Time to first frame is critical
- Scenarios with no swap is common

Let see some system wide optimization and what there benefit and impact. Actually Samsung care much more if not only for the embedded case scenario use of EFL.



Optimization – Reducing Memory Usage Per Application

- Having a software backend helps (GL consumes more than 10MB)
- Letting the kernel choose when to throw away data helps (mmap)
- Reducing memory duplication per process is a first step
  - String share
  - Eet
  - Image/Font cache
  - Copy On Write (When introduced saved 10% of memory and gain 5% on CPU)

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 21

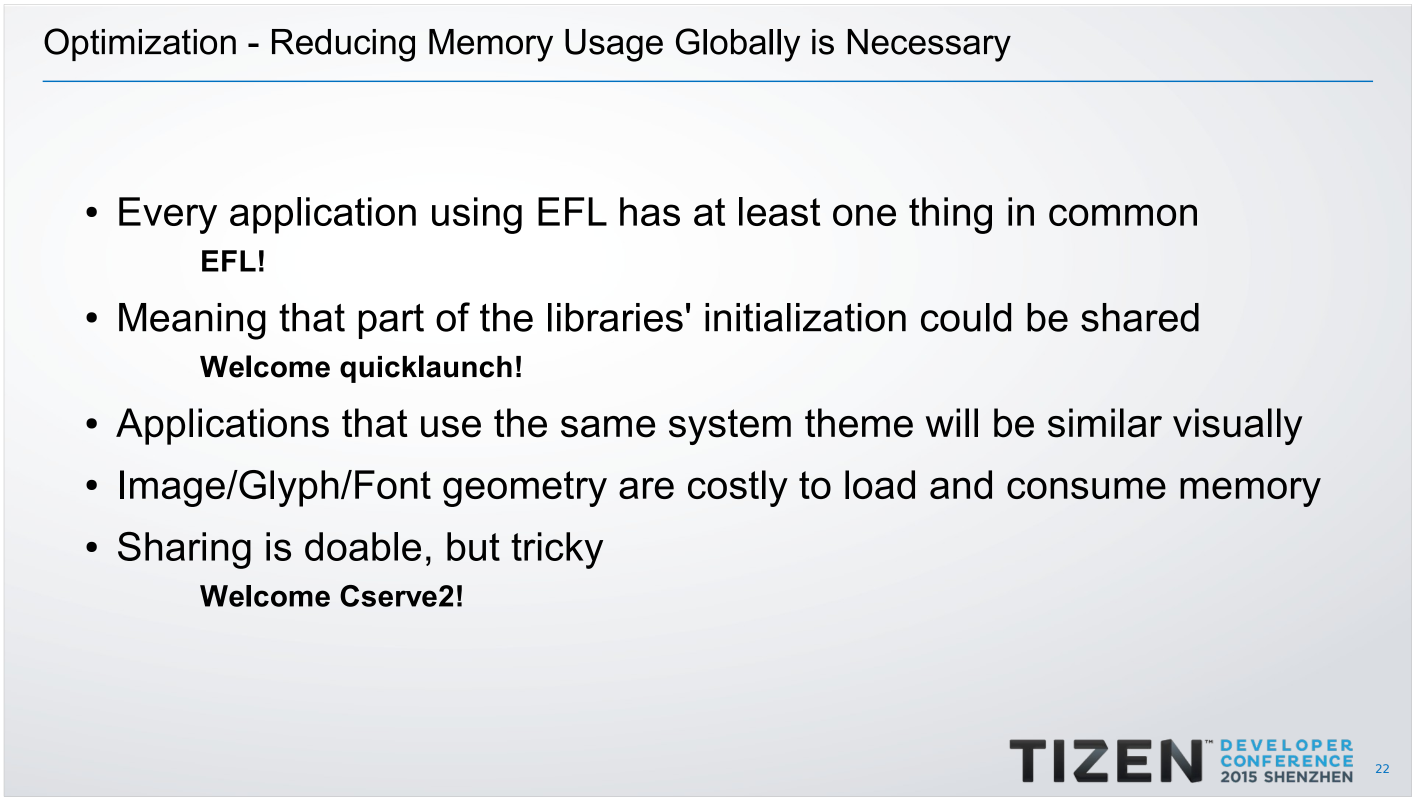
A few trick we use in EFL stack for each application to reduce memory, CPU and battery usage.

- stringshare is just refcounting string in the same process. Speed improvement and memory reduction. Instead of strdup, it does just a +1 on the refcounting !

- Eet share directly data from disk into memory using mmap. This leave the kernel able to remove unused memory optimistically.

- Image/font cache, make sure we do actually reuse the image and font instead of reloading them to often. Also put them as close as possible in memory if possible (with GL backend we do build a texture atlas at runtime to fit all the content that we display).

- Copy On Write (Cow), actually saving memory also give a speed improvement, that's why we did introduce a CoW system in our scenegraph (We should be using it more over time)

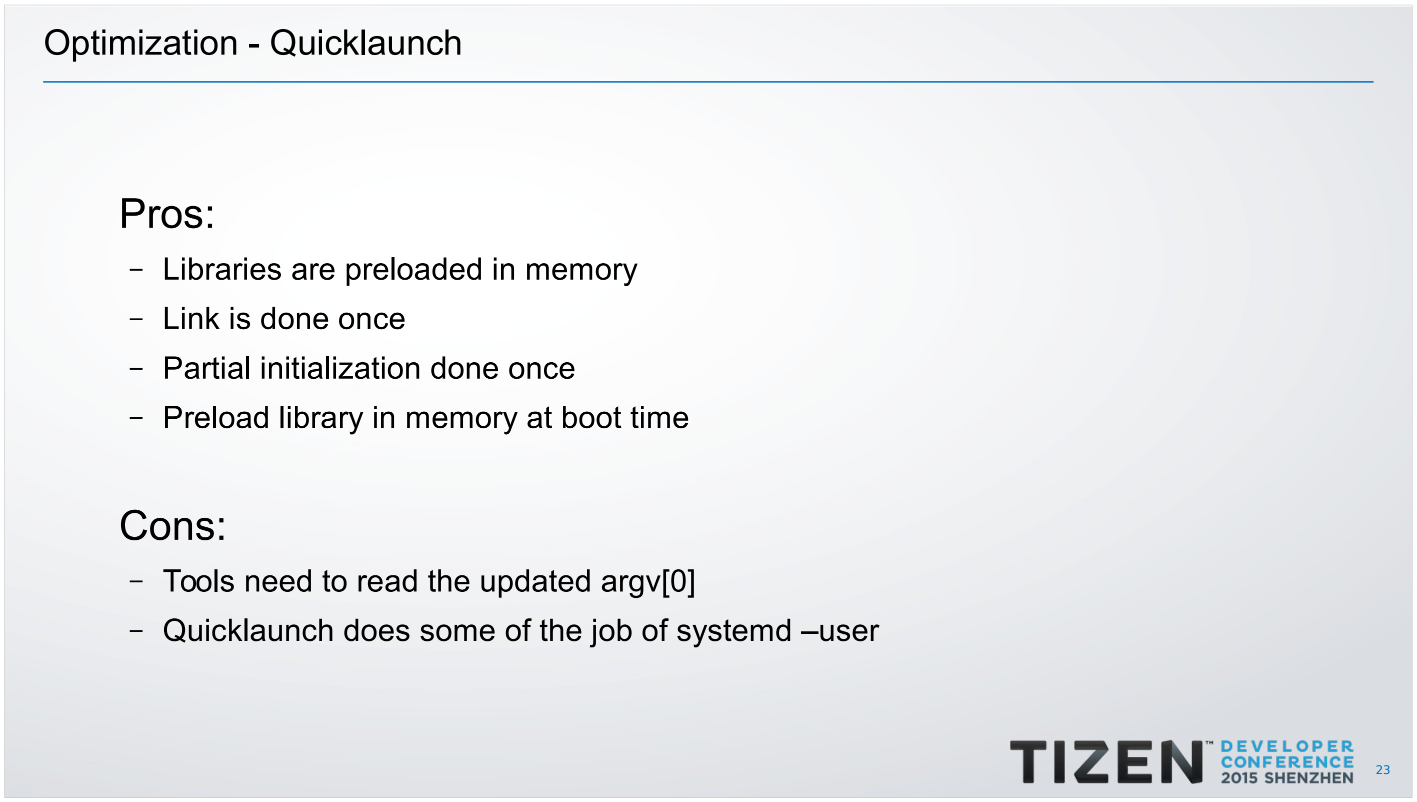


Optimization - Reducing Memory Usage Globally is Necessary

- Every application using EFL has at least one thing in common
  EFL!
- Meaning that part of the libraries' initialization could be shared
  Welcome quicklaunch!
- Applications that use the same system theme will be similar visually
- Image/Glyph/Font geometry are costly to load and consume memory
- Sharing is doable, but tricky
  Welcome Cserve2!

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

Now let's focus on reducing memory duplication accross the system ! What does all application have in common ?!



Optimization - Quicklaunch

Pros:

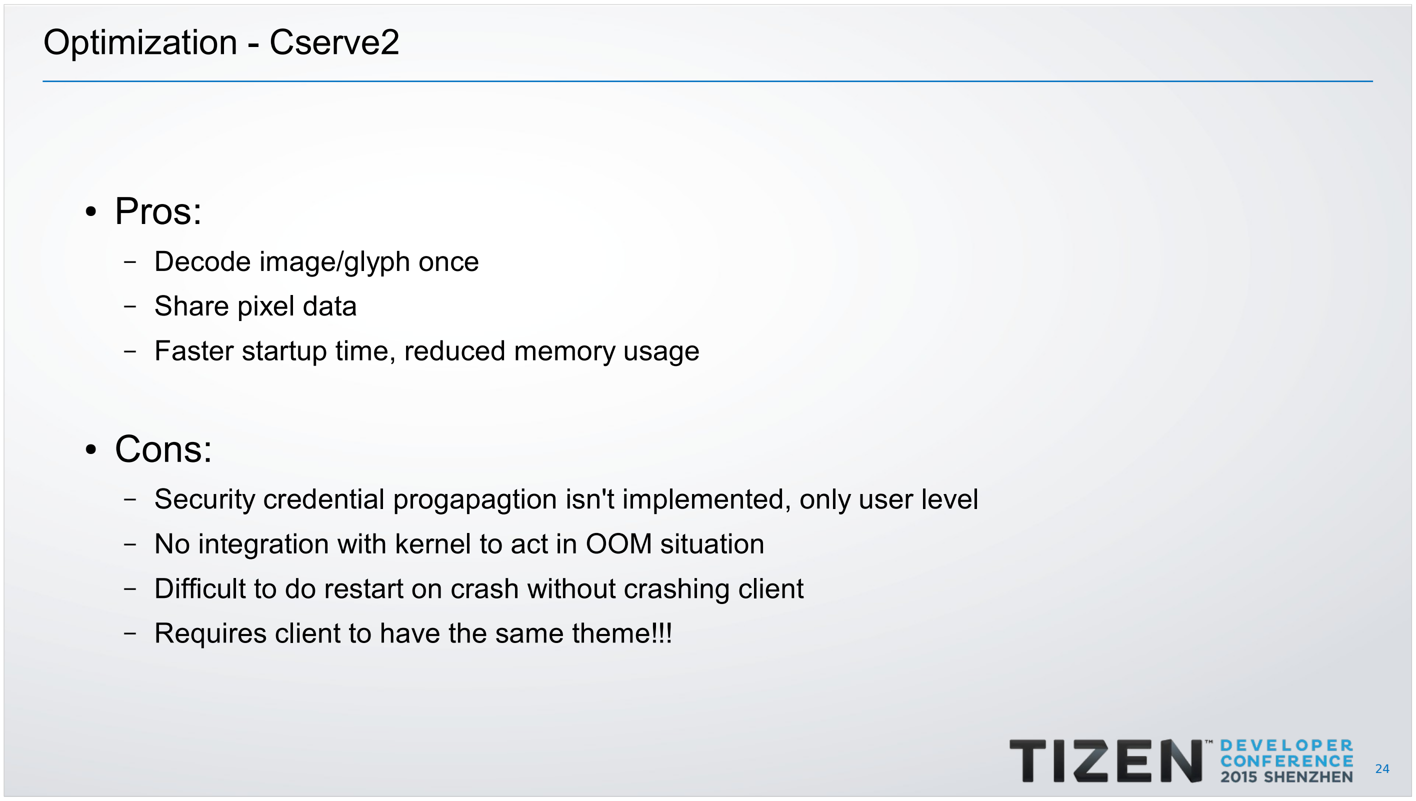
- Libraries are preloaded in memory
- Link is done once
- Partial initialization done once
- Preload library in memory at boot time

Cons:

- Tools need to read the updated argv[0]
- Quicklaunch does some of the job of systemd –user

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 23

So quicklaunch is actually a server that pre start a process linked to EFL and minimally initialize it. Actually in use in Tizen.

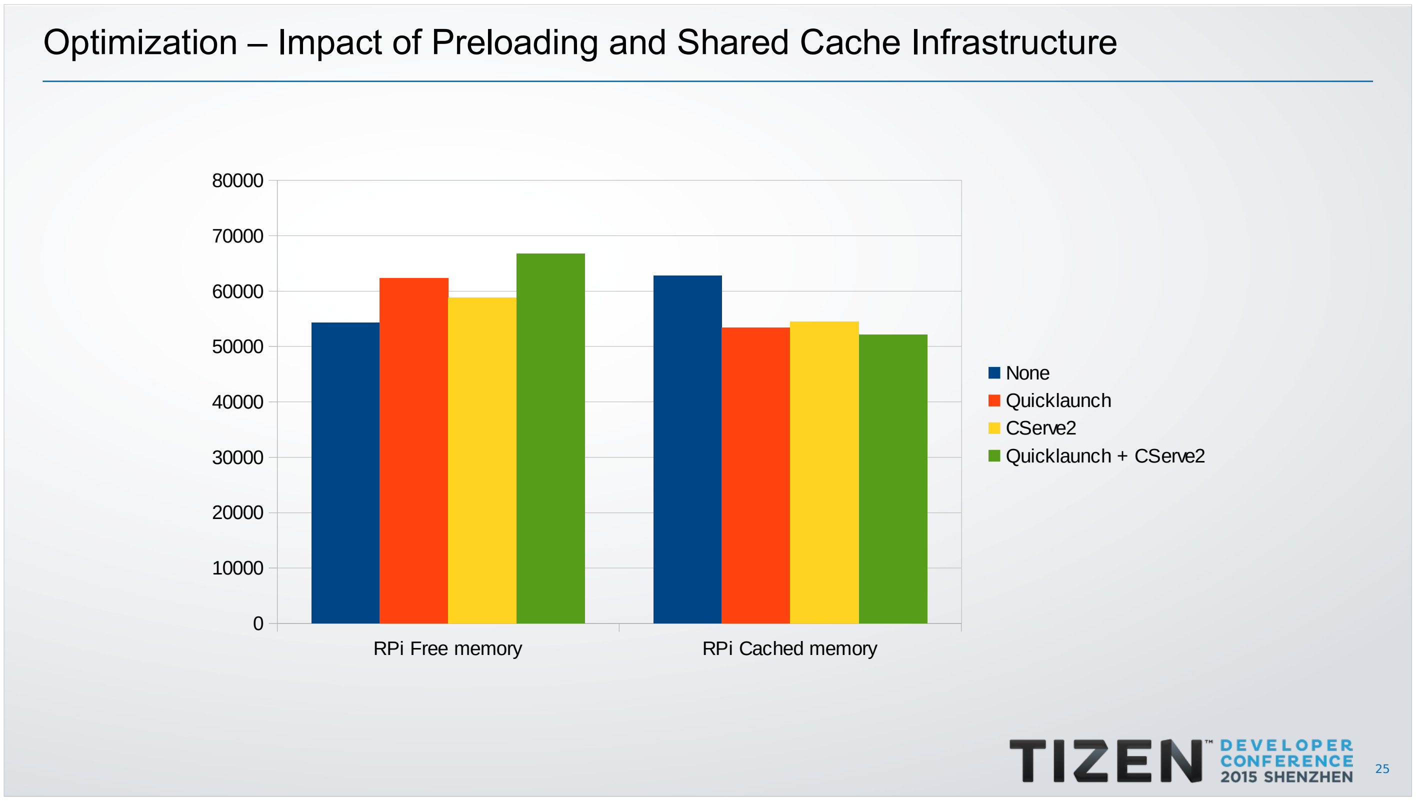


Optimization - Cserve2

- Pros:
  - Decode image/glyph once
  - Share pixel data
  - Faster startup time, reduced memory usage
- Cons:
  - Security credential progapagtion isn't implemented, only user level
  - No integration with kernel to act in OOM situation
  - Difficult to do restart on crash without crashing client
  - Requires client to have the same theme!!!

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

Cserve2 is a server that enable sharing image pixmap, font glyph information and pixmap.



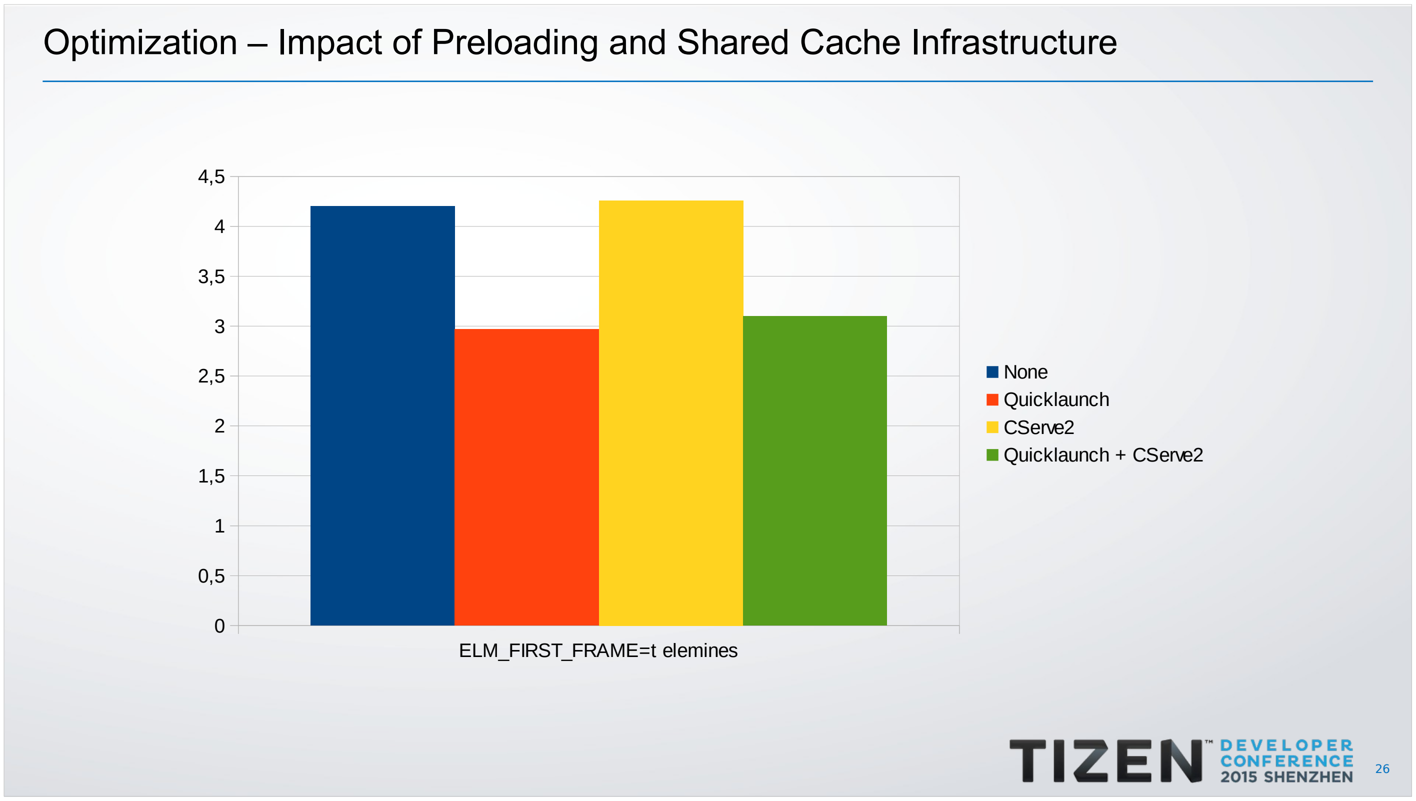
Optimization – Impact of Preloading and Shared Cache Infrastructure

So let's see the impact on free memory and cache memory after a boot.

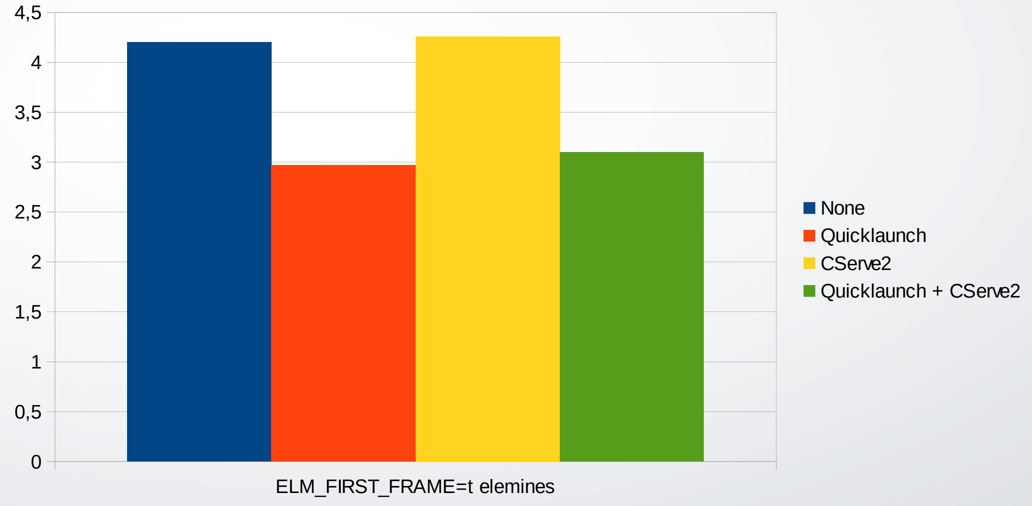
The more free memory the better. It means the system has been able to save some accross the board.

The less cached memory, the less data it did need to load in memory to start. This means usually faster start time. Reducing the cache needed for the system also help giving more room for the kernel for caching more information for application.

As you can see Quicklaunch payoff easily, but actually the combination of Quicklaunch and Cserve2 does pay the most.



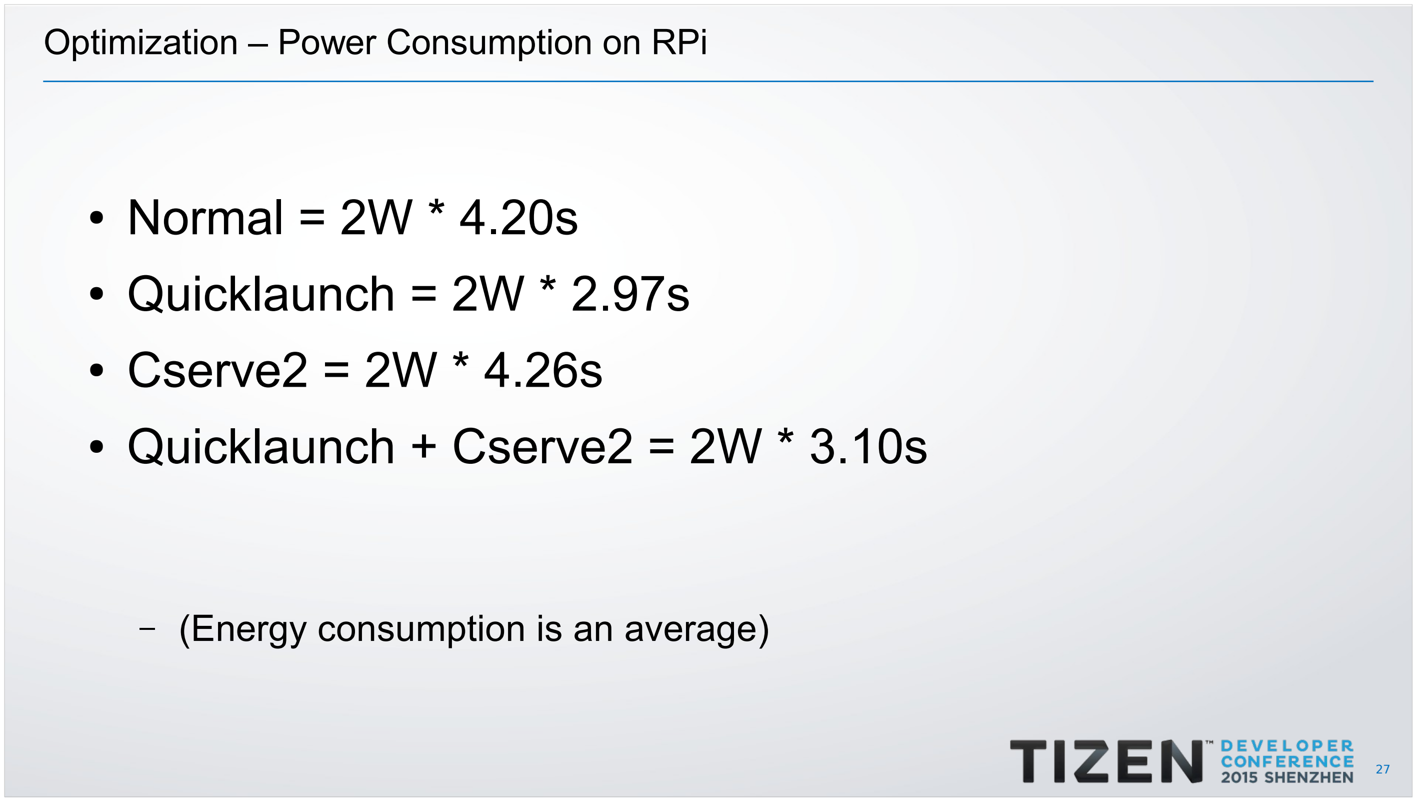
Optimization – Impact of Preloading and Shared Cache Infrastructure



TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 26

Let see the impact of time to first frame on an actual EFL application, elemines (A little swipe mine game).

As you see, only Quicklaunch as a direct benefit. Cserve2 doesn't pay out for application startup yet. This is quite a miss, but there is room for improvement.



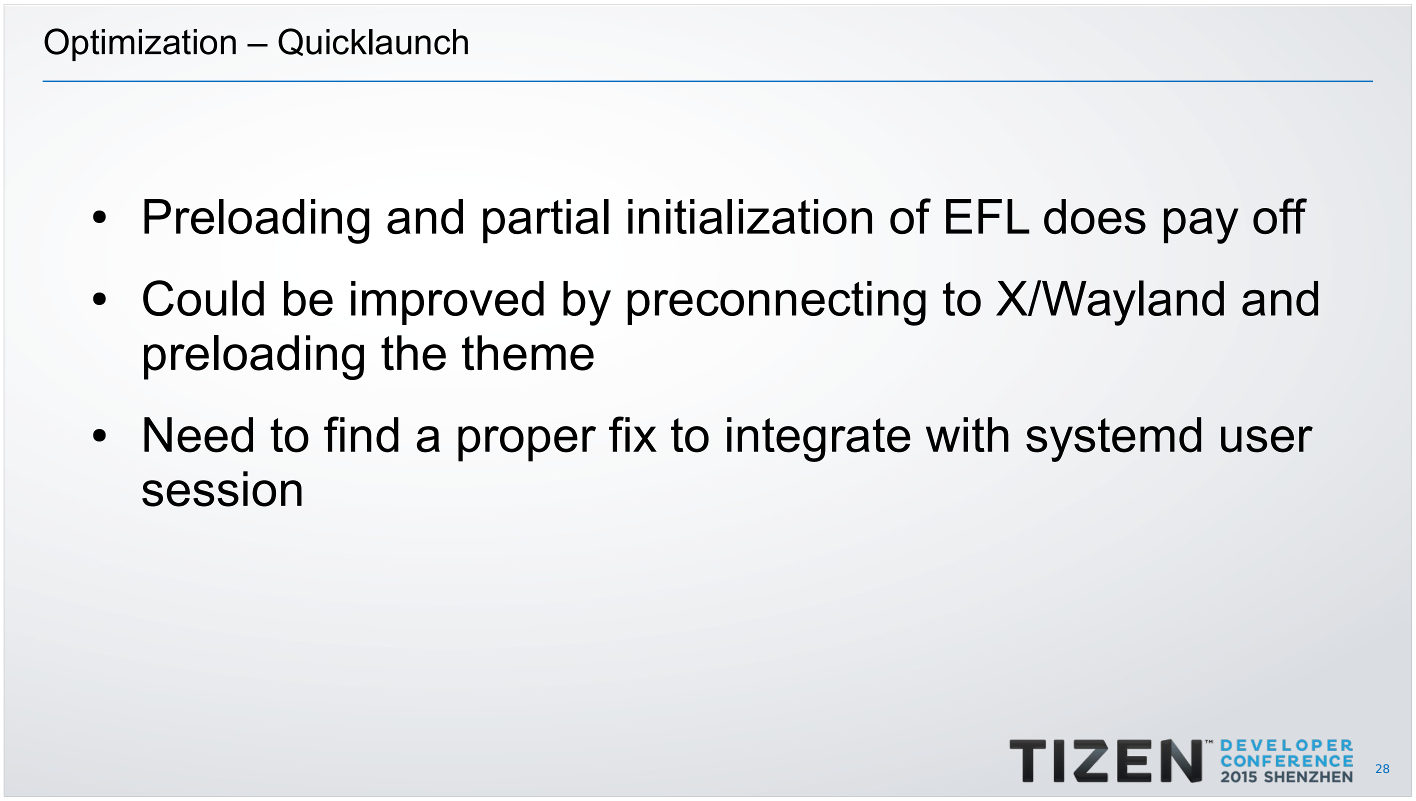
Optimization – Power Consumption on RPi

- Normal = 2W * 4.20s
- Quicklaunch = 2W * 2.97s
- Cserve2 = 2W * 4.26s
- Quicklaunch + Cserve2 = 2W * 3.10s
  - (Energy consumption is an average)

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

27

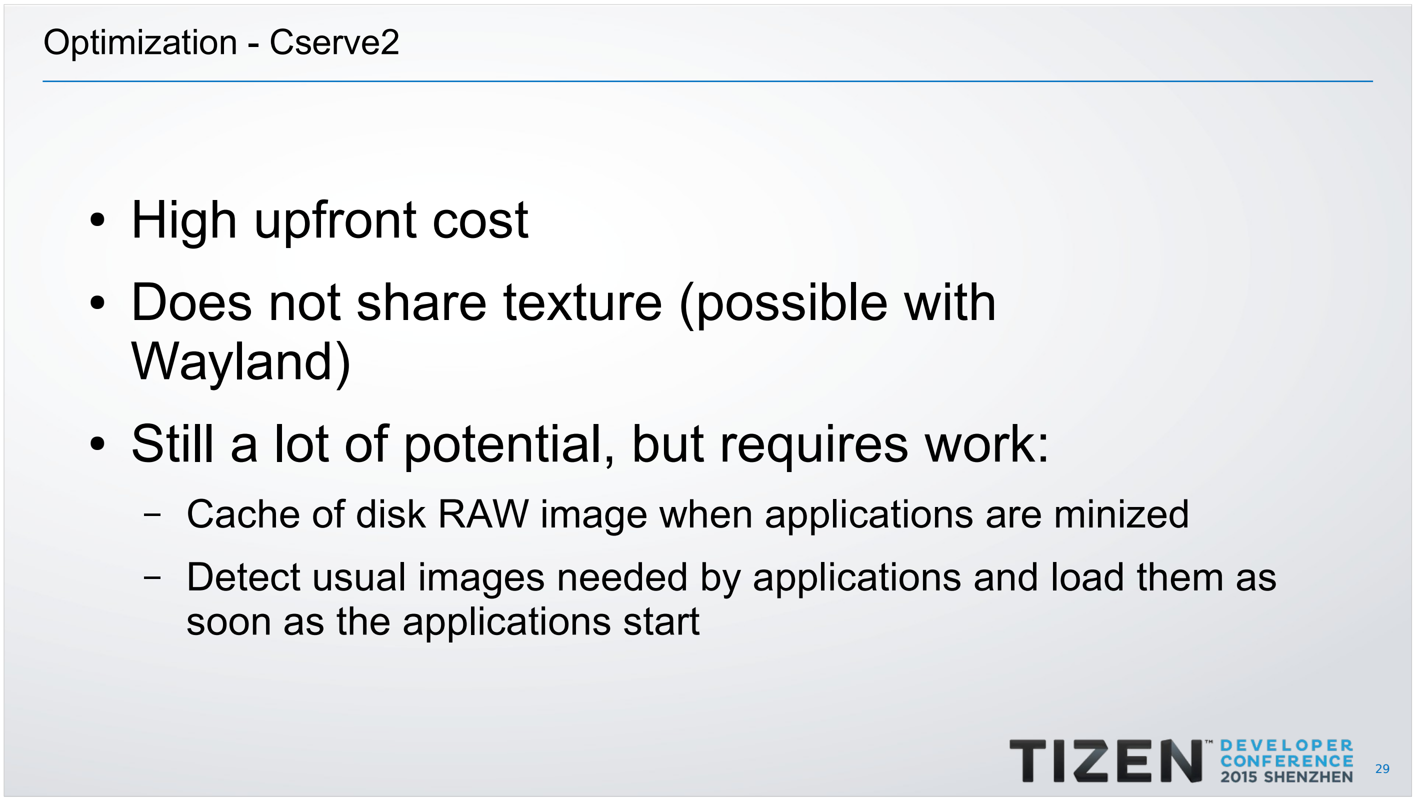
Some energy consumption benchmark here. The idea was to measure the energy consumed for the time to first frame. As you can see, only Quicklaunch is really worth it at this stage (Which explain it's use in Tizen)



Optimization – Quicklaunch

- Preloading and partial initialization of EFL does pay off
- Could be improved by preconnecting to X/Wayland and preloading the theme
- Need to find a proper fix to integrate with systemd user session

So let make some conclusion on quicklaunch.

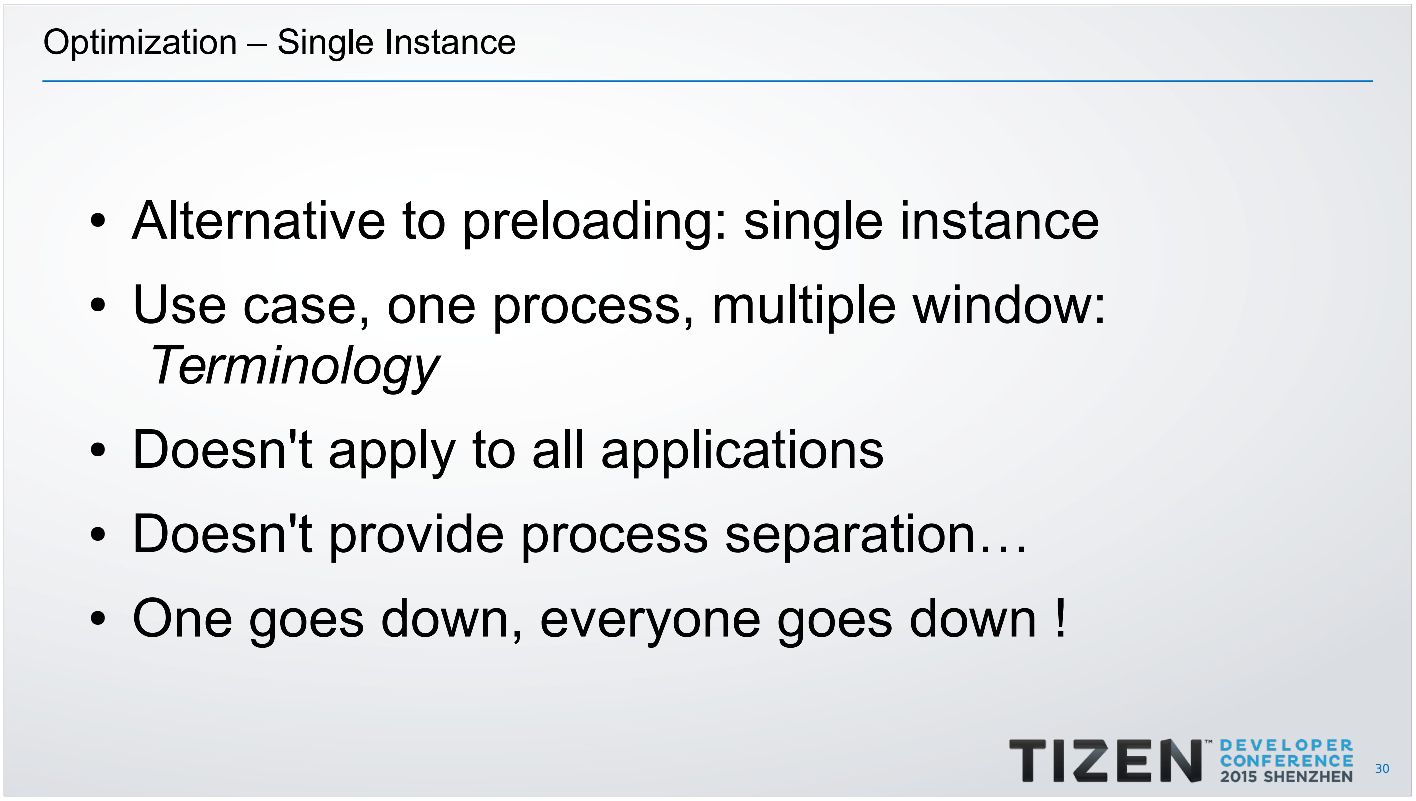


Optimization - Cserve2

- High upfront cost
- Does not share texture (possible with Wayland)
- Still a lot of potential, but requires work:
  - Cache of disk RAW image when applications are minized
  - Detect usual images needed by applications and load them as soon as the applications start

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

And now let's conclude about Cserve2.

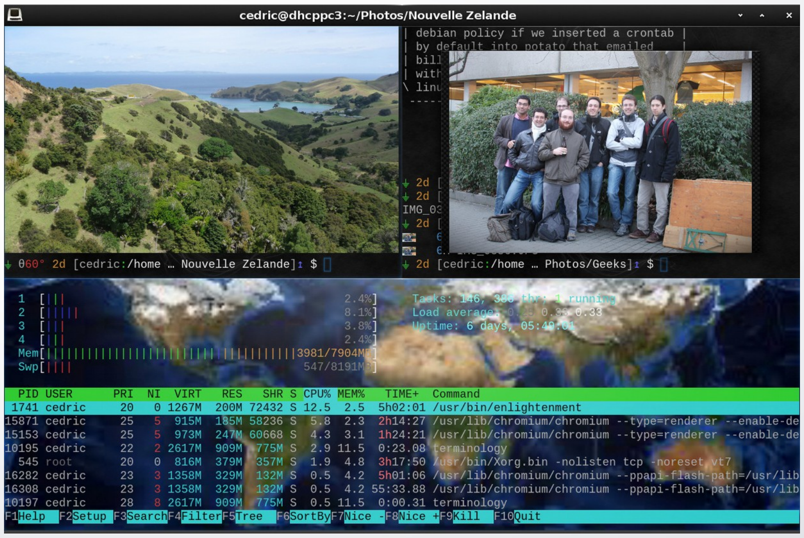

Optimization – Single Instance

- Alternative to preloading: single instance
- Use case, one process, multiple window: *Terminology*
- Doesn't apply to all applications
- Doesn't provide process separation…
- One goes down, everyone goes down !

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

Let's look at an alternate solution, instead of having multiple process that duplicate ressource, let's have just one with multiple example !

Terminology



```
| debian policy if we inserted a crontab |
| by default into potato that emailed    |
| bill
| with
\ linu
 -----

2d [
2d [
IMG_03
2d [
```

```
θ60° 2d [cedric:/home … Nouvelle Zelande]↕ $
```

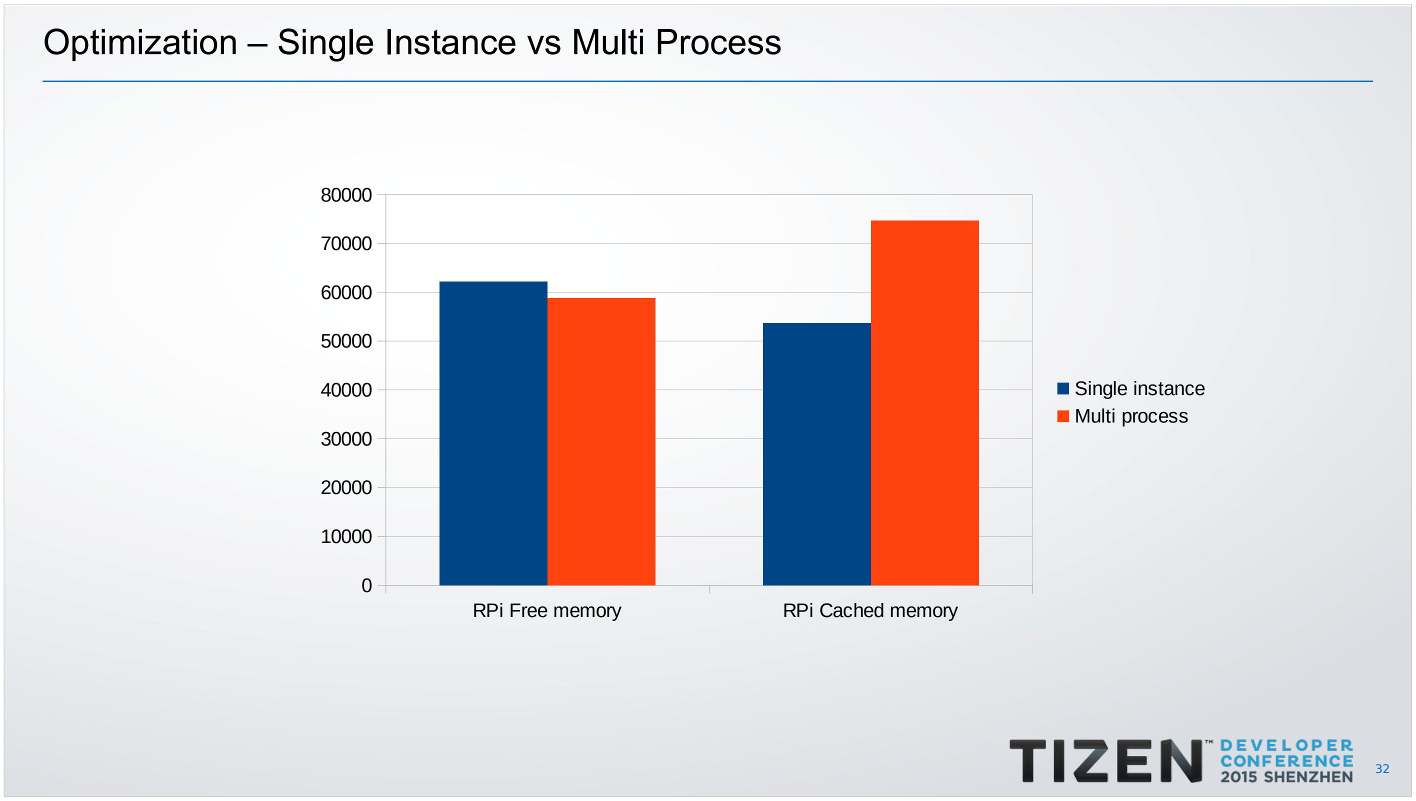
```
2d [cedric:/home … Photos/Geeks]↕ $
```

```
  1  [|||                                   2.4%]     Tasks: 146, 388 thr; 1 running
  2  [|||||                                 8.1%]     Load average: 0.25 0.33 0.33
  3  [|||                                   3.8%]     Uptime: 6 days, 05:40:01
  4  [|||                                   2.4%]
  Mem[||||||||||||||||||||||||||||||||||||3981/7904MB]
  Swp[||||                               547/8191MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 1741 cedric     20   0 1267M  200M 72432 S 12.5  2.5  5h02:01 /usr/bin/enlightenment
15871 cedric     25   5  915M  185M 58236 S  5.8  2.3  2h14:27 /usr/lib/chromium/chromium --type=renderer --enable-de
15153 cedric     25   5  973M  247M 60668 S  4.3  3.1  1h24:21 /usr/lib/chromium/chromium --type=renderer --enable-de
10195 cedric     22   2 2617M  909M  775M S  2.9 11.5  0:23.08 terminology
  545 root       20   0  816M  379M  357M S  1.9  4.8  3h17:50 /usr/bin/Xorg.bin -nolisten tcp -noreset vt7
16282 cedric     23   3 1358M  329M  132M S  0.5  4.2  5h01:06 /usr/lib/chromium/chromium --ppapi-flash-path=/usr/lib
16308 cedric     23   3 1358M  329M  132M S  0.5  4.2 55:33.88 /usr/lib/chromium/chromium --ppapi-flash-path=/usr/lib
10197 cedric     28   8 2617M  909M  775M S  0.5 11.5  0:00.31 terminology
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

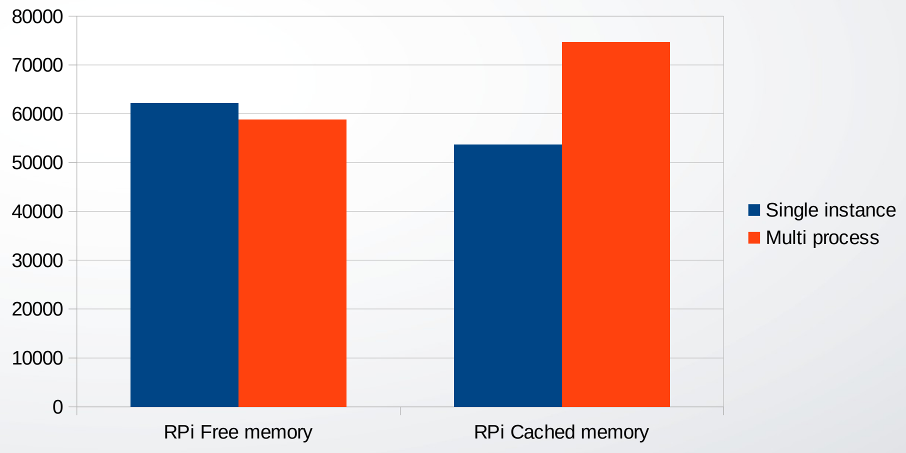
TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

As an example we will be taking Terminology an EFL based replacement for xterm and any kind of terminal console emulator, but much more modern !

Able to display image and video, play sound, split screen and have multiple tab… Oh and as fast as urxvt the reference for terminal emulator, but much more prettier !

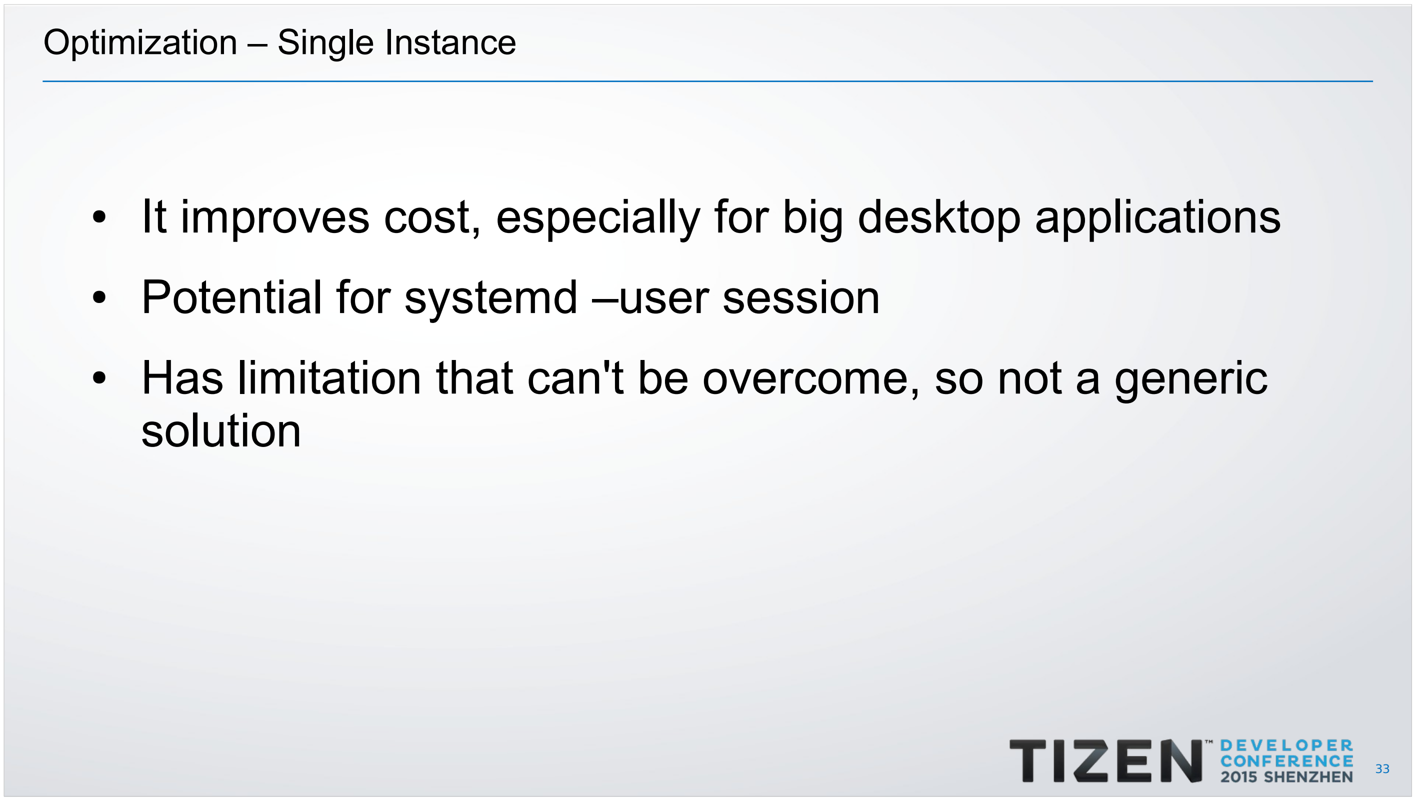


Optimization – Single Instance vs Multi Process



TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

As you can see, we do have a win on free memory, that is not bad, but cached memory, strangely not. I don't have an explanation that satisfy me for that case, and this require some more analyzing.



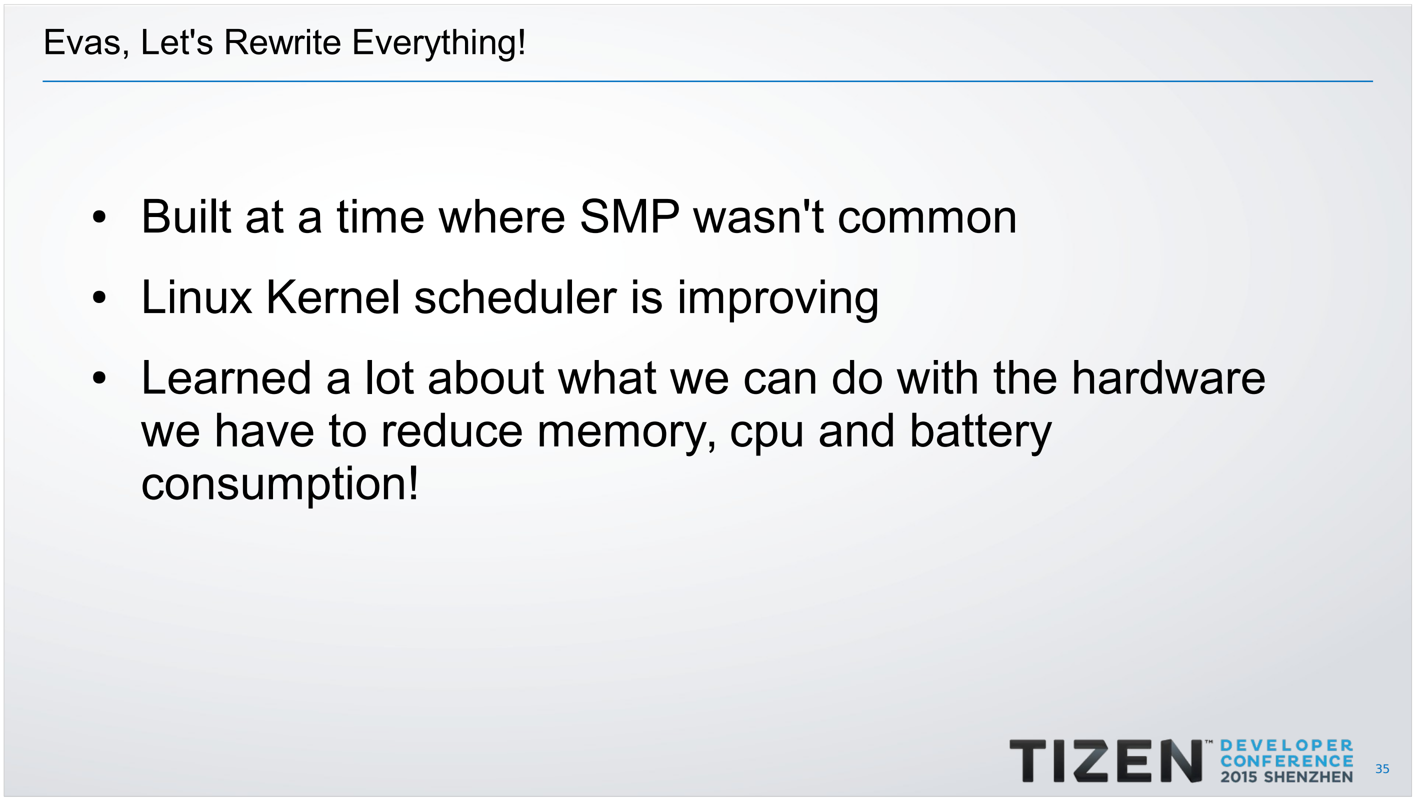
Optimization – Single Instance

- It improves cost, especially for big desktop applications
- Potential for systemd –user session
- Has limitation that can't be overcome, so not a generic solution

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

Let's conclude this topic !

- EFL community cares about performance
- Samsung cares about performance for all product lines
- There is always room for improvement
- Evas scenegraph logic is more than 10 years old, refactoring and cleaning is necessary
- Improving our tests suites and especially our benchmark is a permanent task
- Never ending story! Always something to improve!

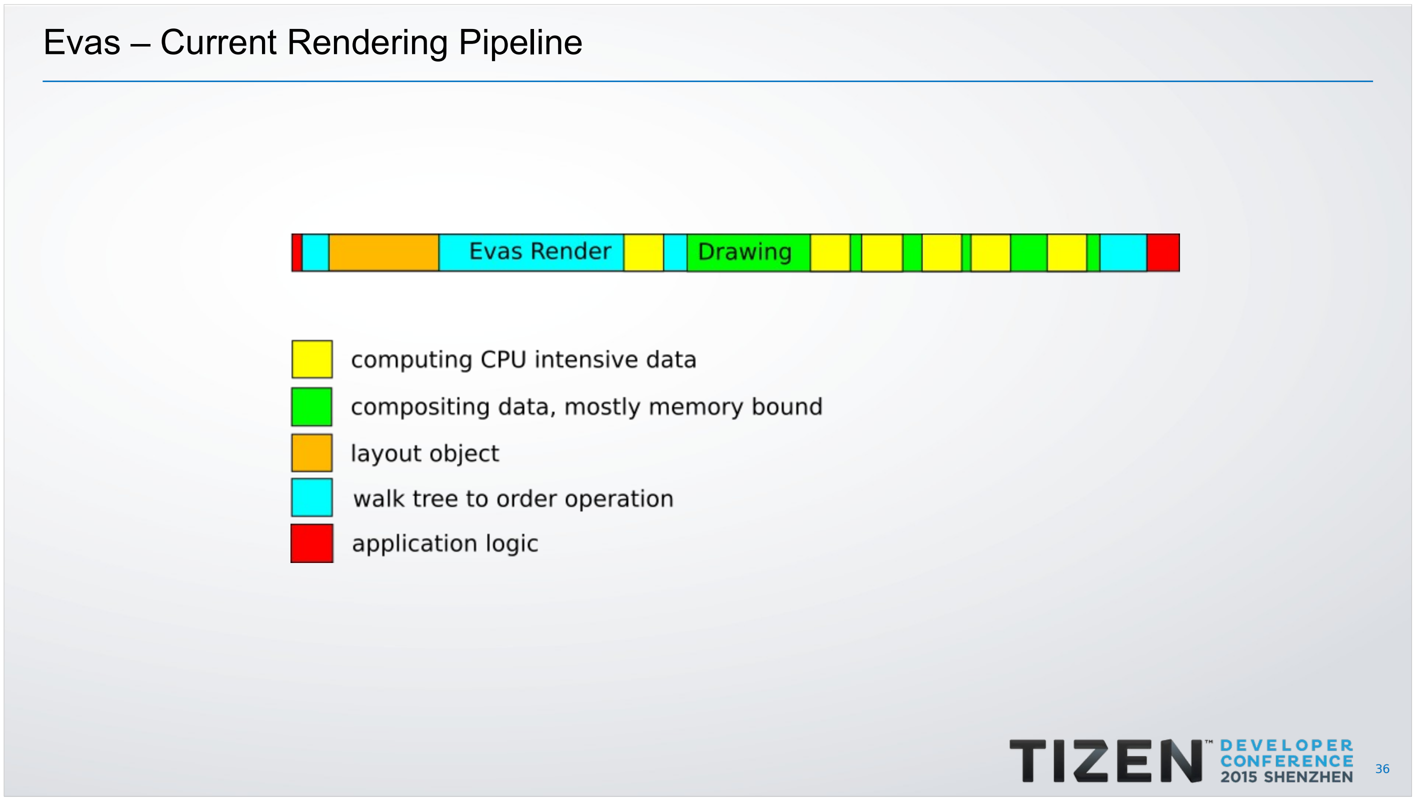


Evas, Let's Rewrite Everything!

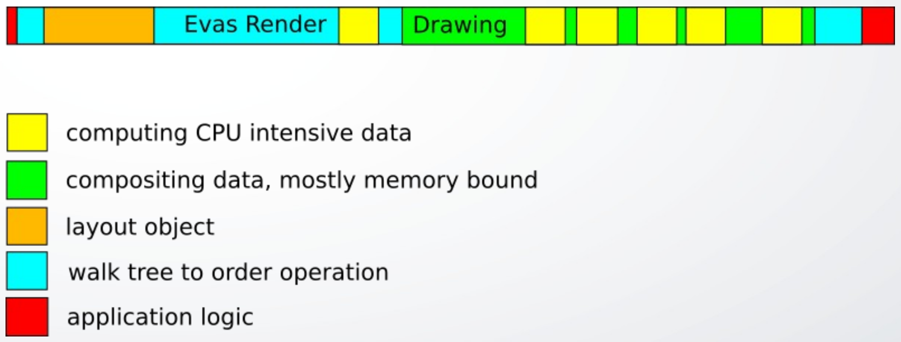
- Built at a time where SMP wasn't common
- Linux Kernel scheduler is improving
- Learned a lot about what we can do with the hardware we have to reduce memory, cpu and battery consumption!

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN 35

There is a lot of things going on with the kernel scheduler that I don't have time to talk about here, but this has serious real impact on application performance and battery usage. Don't hesitate to come and talk with me on that subject later.



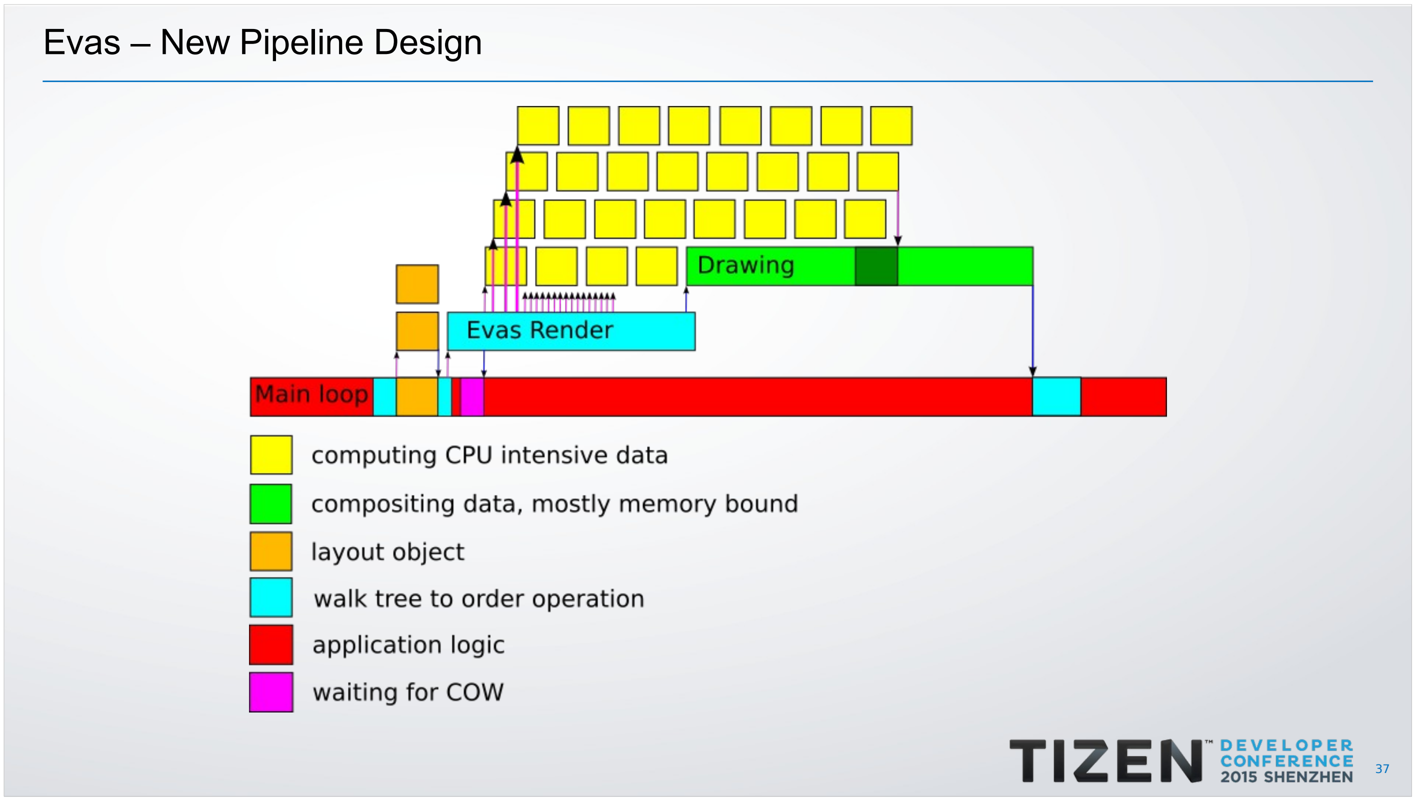
Evas – Current Rendering Pipeline


Evas Render
Drawing
computing CPU intensive data
compositing data, mostly memory bound
layout object
walk tree to order operation
application logic

So the current rendering pipeline of Evas for GL is like this. For software backend, all the green and yellow in between now happen in a thread, leaving more room for the main loop application code to run.

As you can see, the profile of the main thread is changing quickly over time and not much time is left to the application. Which should not be a serious matter as most application do not need to do anything heavy as that is what the toolkit does for them, but problem is that the kernel is super confused by this behavior and always bad at setting the right clock speed and number of core on.

Actually this doesn't also fit with modern design like Vulkan enable.

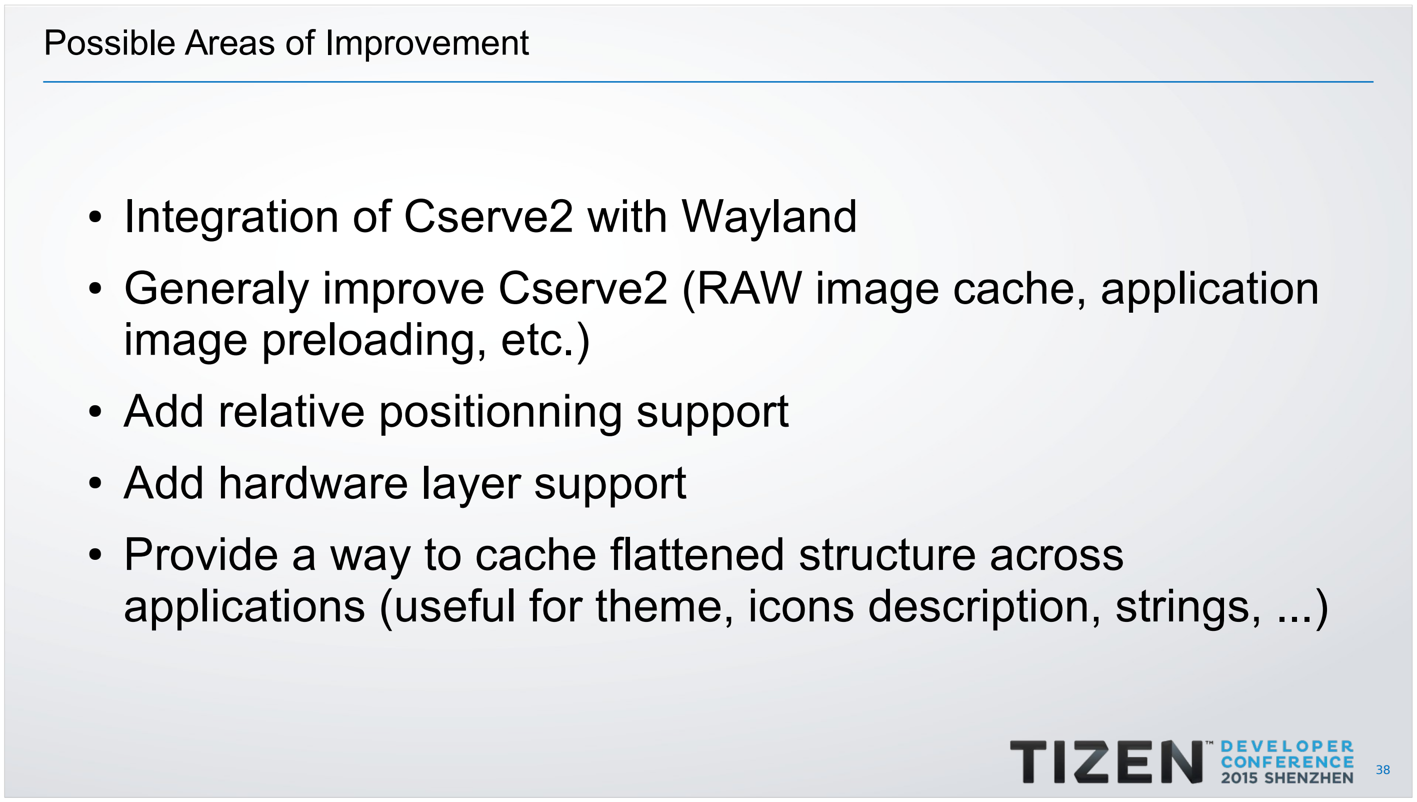


Evas – New Pipeline Design

🟨	computing CPU intensive data
🟩	compositing data, mostly memory bound
🟧	layout object
🟦	walk tree to order operation
🟥	application logic
🟪	waiting for COW

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

This is where we are slowly going. Making sure that the kernel understand what each thread is doing and going to do, scaling better with coming Vulkan. That is a very massive amount of work to do and is going to likely take another year before it is « done », but as we do time base release, we will slowly bring improvement in.



Possible Areas of Improvement

- Integration of Cserve2 with Wayland
- Generaly improve Cserve2 (RAW image cache, application image preloading, etc.)
- Add relative positionning support
- Add hardware layer support
- Provide a way to cache flattened structure across applications (useful for theme, icons description, strings, ...)

TIZEN™ DEVELOPER CONFERENCE 2015 SHENZHEN

If people want to join, there is a lot of place where we can still improve EFL and where I can help mentor them to contribute there.

There's Still a Lot of Work to Do!

- Contributions are welcome from all around the world!
- True, open source project with a community; everyone can c
ontribute directly!
- More companies are using it other than Samsung
- Join the mailing list and IRC!
- And poke me around any time during the event if you have question!

Questions?