Desktops on a Diet

Carsten Haitzler <raster@rasterman.com>
Enlightenment
http://www.enlightenment.org

Todays “Average” PC

- $1,210 (2005 USA “Average” PC – Gartner)
- $190 (OLPC Laptop 366Mhz x86, 128M RAM)

The Average Income (Australia)

- $43,600
- 36 Average PC's ($1,210)

Average income in China

- $1,635
- 1.35 Average PC's ($1,210)

What a PC costs in China

- Average PC costs the “equivalent” of $32,290
- OLPC “equivalent cost” is $5,060

Very unhappy users

- If you went out tomorrow and spent $5000 on a new PC and it couldn't run a Linux desktop – you would call it ridiculous
- If you had to spend $32,000 to get it to run – you would be annoyed
- There needs to be more attention to trying to be efficient
- It matters to people who are not as obscenely rich as us

A comparison of Desktops

- GNOME 2.16.1
- KDE 3.5.5
- XFCE 4.3.99
- Enlightenment 0.16.999.037 (E17)

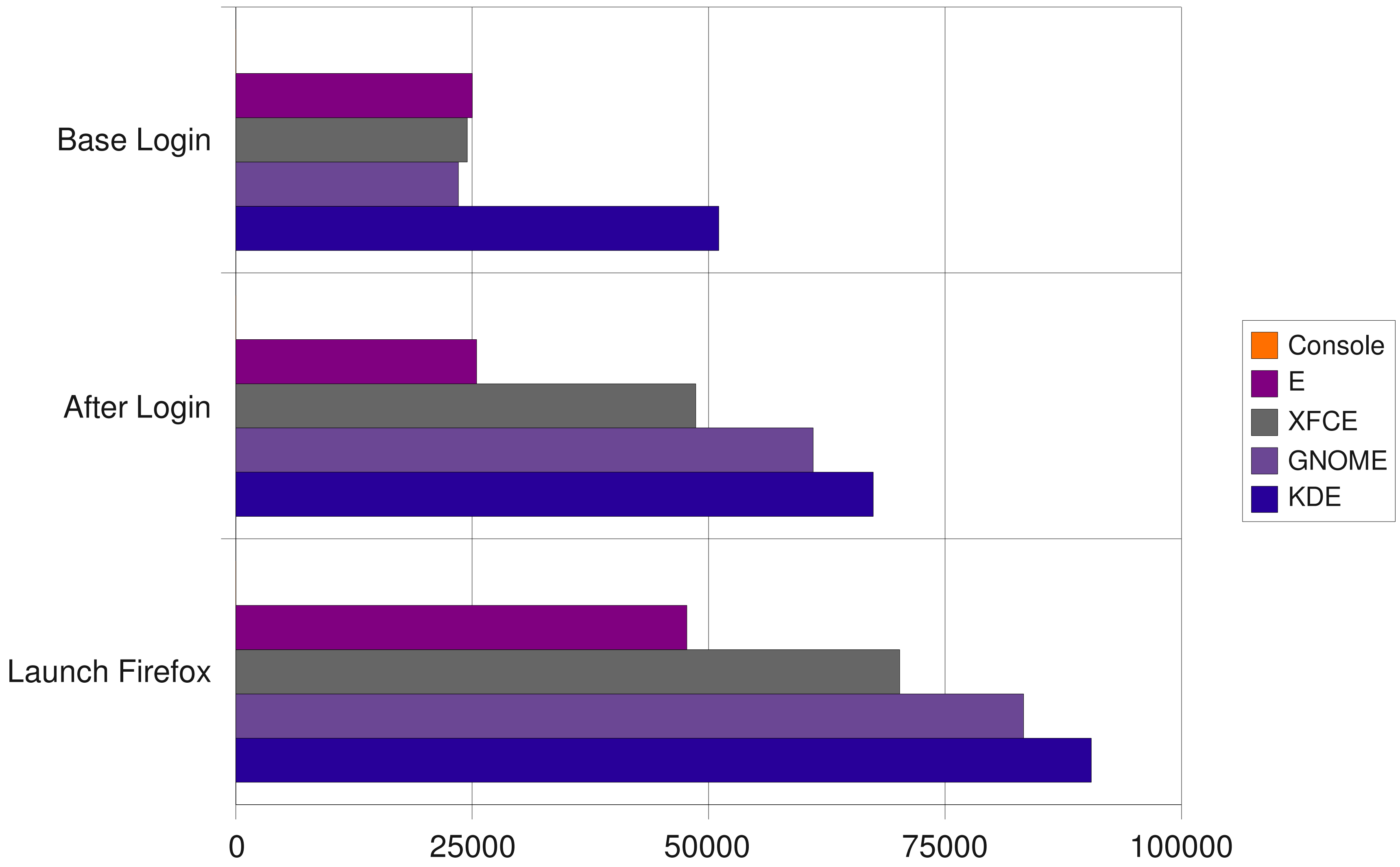
Comparisons of Desktops

	Absolute memory usage from fresh boot				
	Memory Usage Only (Kb)				
	Console	**E**	**XFCE**	**GNOME**	**KDE**
Base Login	51608	76632	76088	75156	102680
After Login	51608	77096	100264	112668	119028
Launch Firefox	51608	99324	121816	134924	142076
	Files IO Access (Kb)				
	Console	**E**	**XFCE**	**GNOME**	**KDE**
Base Login	90368	103364	103500	102360	144880
After Login	90368	108308	173476	164672	201908
Launch Firefox	90368	136752	195124	188408	228280

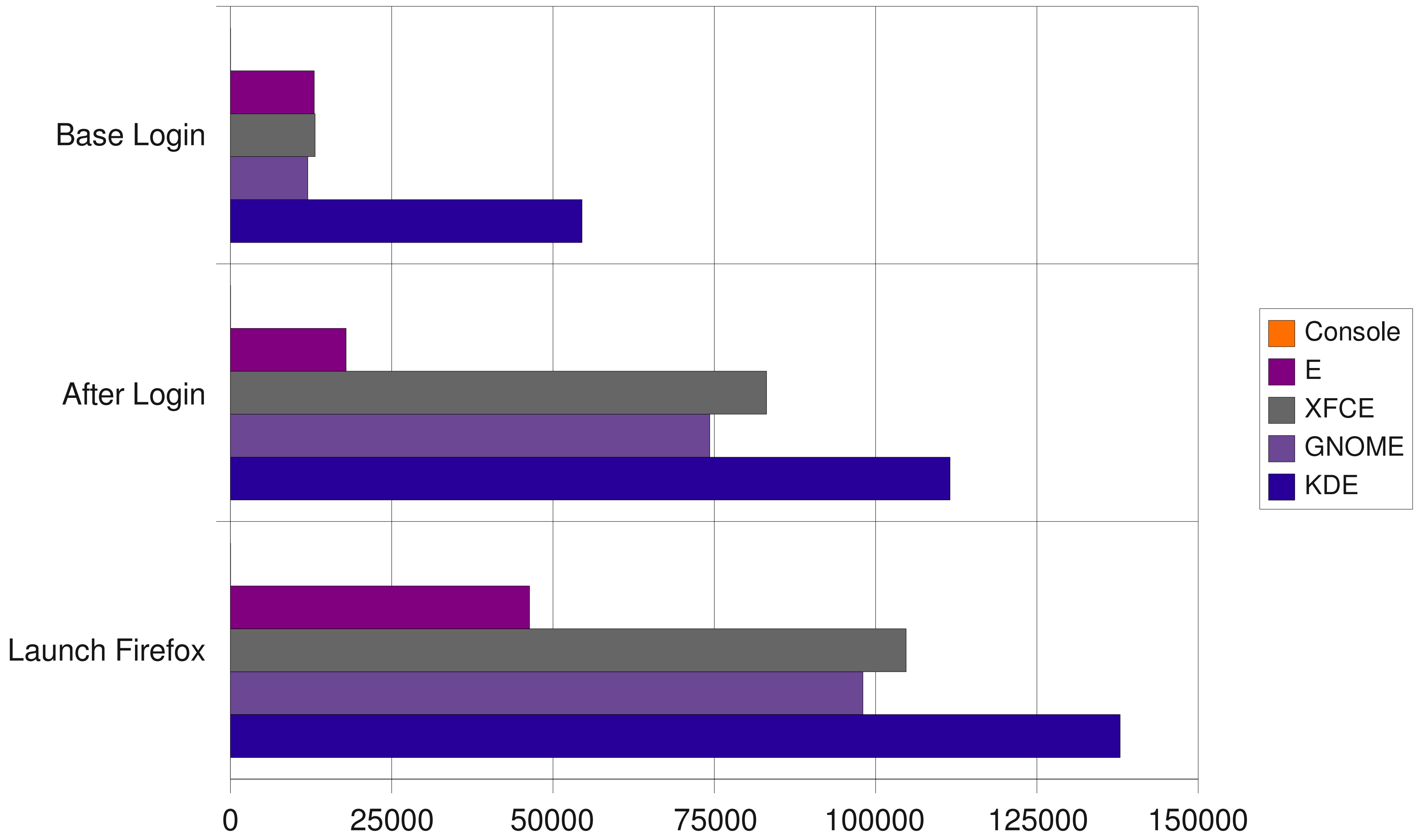
Relative Memory needs

	Relative memory usage compared to base console system				
	Memory Usage Only (Kb) Relative to Console				
	Console	E	XFCE	GNOME	KDE
Base Login	0	25024	24480	23548	51072
After Login	0	25488	48656	61060	67420
Launch Firefox	0	47716	70208	83316	90468
	Files IO Access (Kb) Relative to Console				
	Console	E	XFCE	GNOME	KDE
Base Login	0	12996	13132	11992	54512
After Login	0	17940	83108	74304	111540
Launch Firefox	0	46384	104756	98040	137912

Relative Memory needs

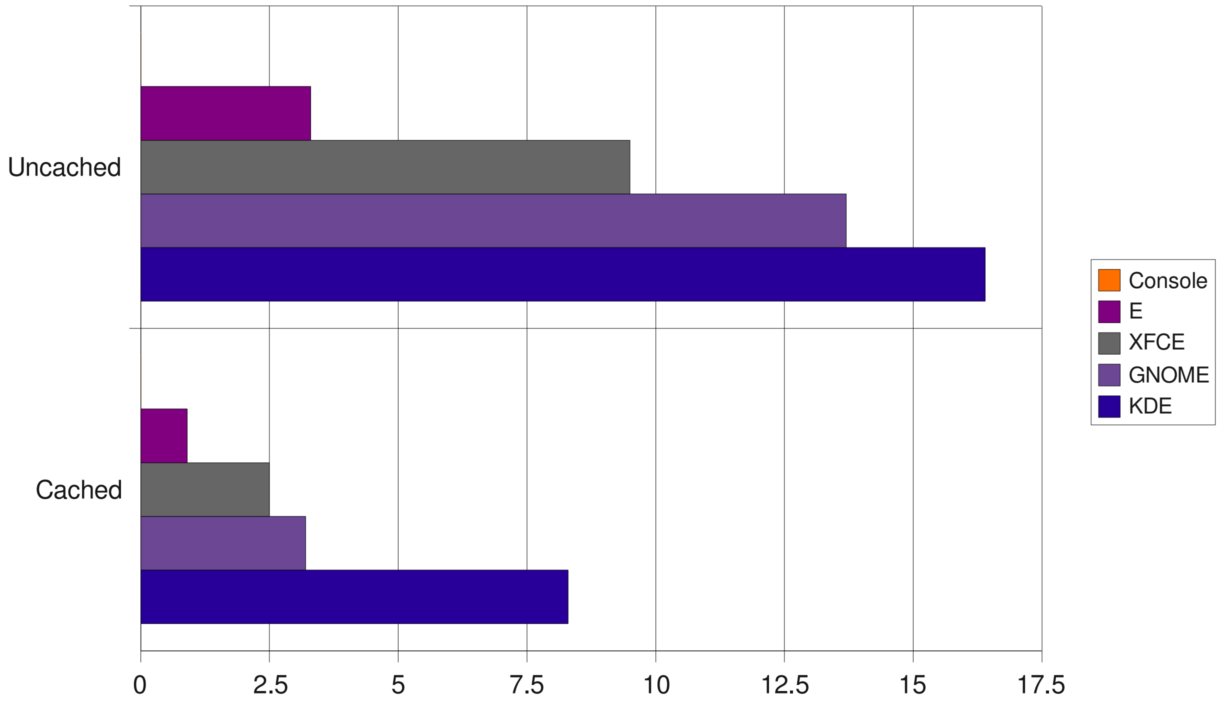


Relative Disk IO Usage



Login Times

	Login time (Seconds)				
	Console	**E**	**XFCE**	**GNOME**	**KDE**
Uncached	0	3.3	9.5	13.7	16.4
Cached	0	0.9	2.5	3.2	8.3



How do you get there?

- Care about people with lesser machines
- Do statistics and analysis
- Investigate techniques used elsewhere
- Think carefully about your designs
- Here are some things used for Enlightenment development to get there

Time your code

```
ESTART: 0.00000 [0.00000] - begin
ESTART: 0.00015 [0.00014] - signals done
ESTART: 0.20644 [0.20630] - determine prefix
ESTART: 0.21657 [0.01013] - prefix done
ESTART: 0.21664 [0.00007] - intl init
ESTART: 0.21813 [0.00150] - parse args
ESTART: 0.21816 [0.00003] - arg parse done
ESTART: 0.64135 [0.42318] - edje init
ESTART: 0.64162 [0.00028] - ecore init
ESTART: 0.64179 [0.00017] - ecore_file init
ESTART: 0.64194 [0.00015] - more ecore
ESTART: 0.64198 [0.00004] - x connect
...
ESTART: 1.74095 [0.00001] - load modules
ESTART: 2.01361 [0.27267] - gadcon
ESTART: 2.01364 [0.00003] - shelves
ESTART: 2.01366 [0.00001] - exebuf
ESTART: 2.01368 [0.00002] - desklock
ESTART: 2.01388 [0.00020] - add idle enterers
ESTART: 2.01452 [0.00064] - init properites
ESTART: 2.28726 [0.27274] - test code
ESTART: 2.28730 [0.00003] - shelf config init
ESTART: 3.47224 [1.18494] - MAIN LOOP AT LAST
ESTART: 3.57884 [0.10660] - SLEEP
```

How did this help?

- Removed useless X round-trips
- Removed pointless init code
- Allowed benchmarking when implementing disk pre-caching

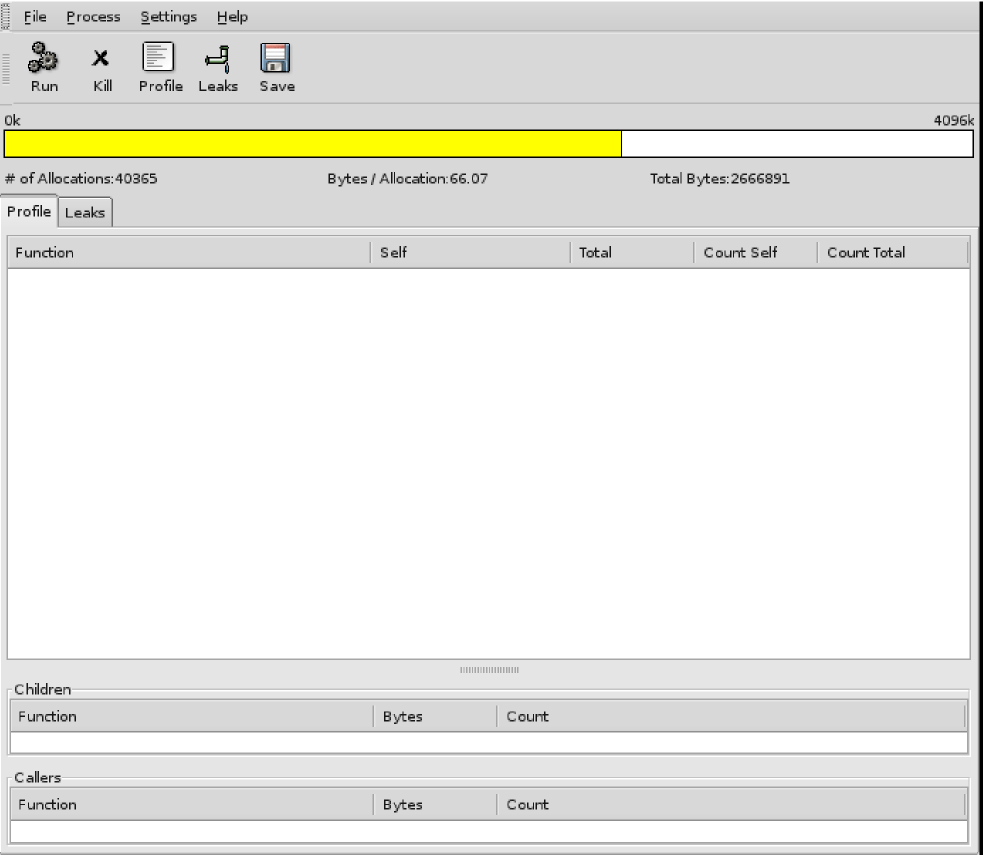
Pre-caching

- A technique several OS's use to pre-fetch data from disk you probably will need
- Implemented as an LD_PRELOAD and a logging mechanism, with replay
- Shaved uncached startup time in half once implemented
- Currently is extremely naïve and could be much smarter with kernel help

Memprof

- Little known tool
- Tells you in great detail who allocated what memory and where and how much
- Helped identify lots of empty string (1 byte) allocations that we removed with a string sharing subsystem
- Recently has started development again

Metacity Memory Use



# of Allocations: 40365   Bytes / Allocation: 66.07   Total Bytes: 2666891

Function	Self	Total	Count Self	Count Total

Children

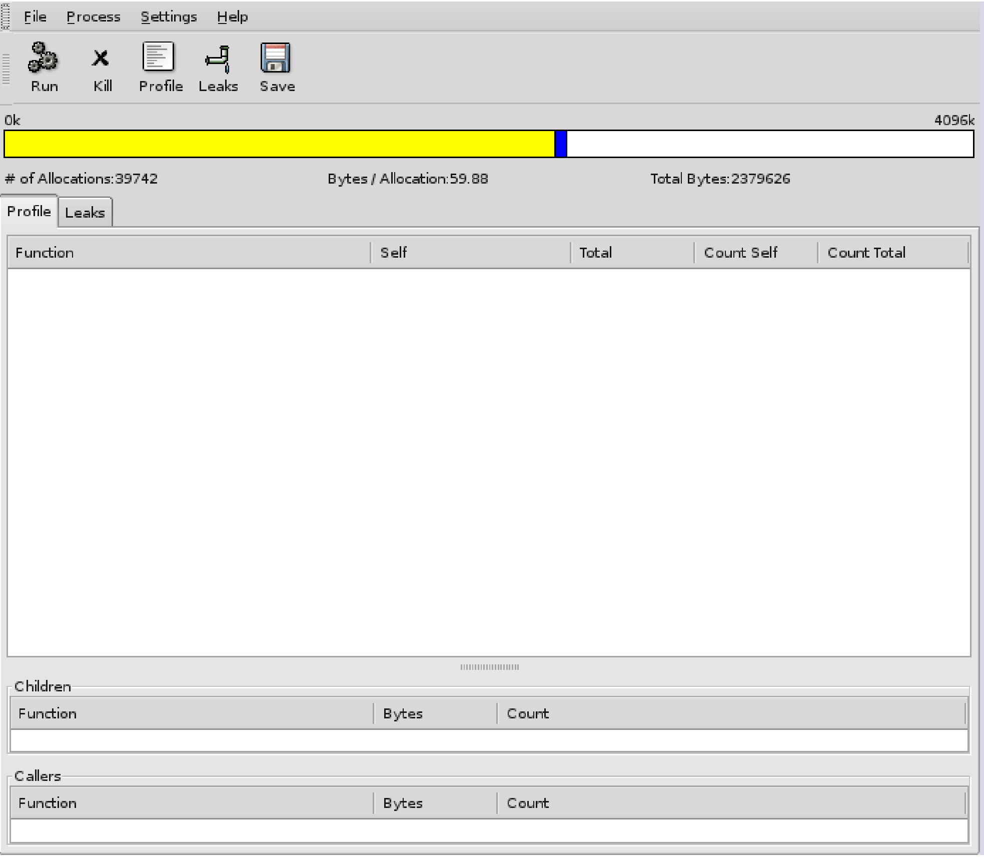
Function	Bytes	Count

Callers

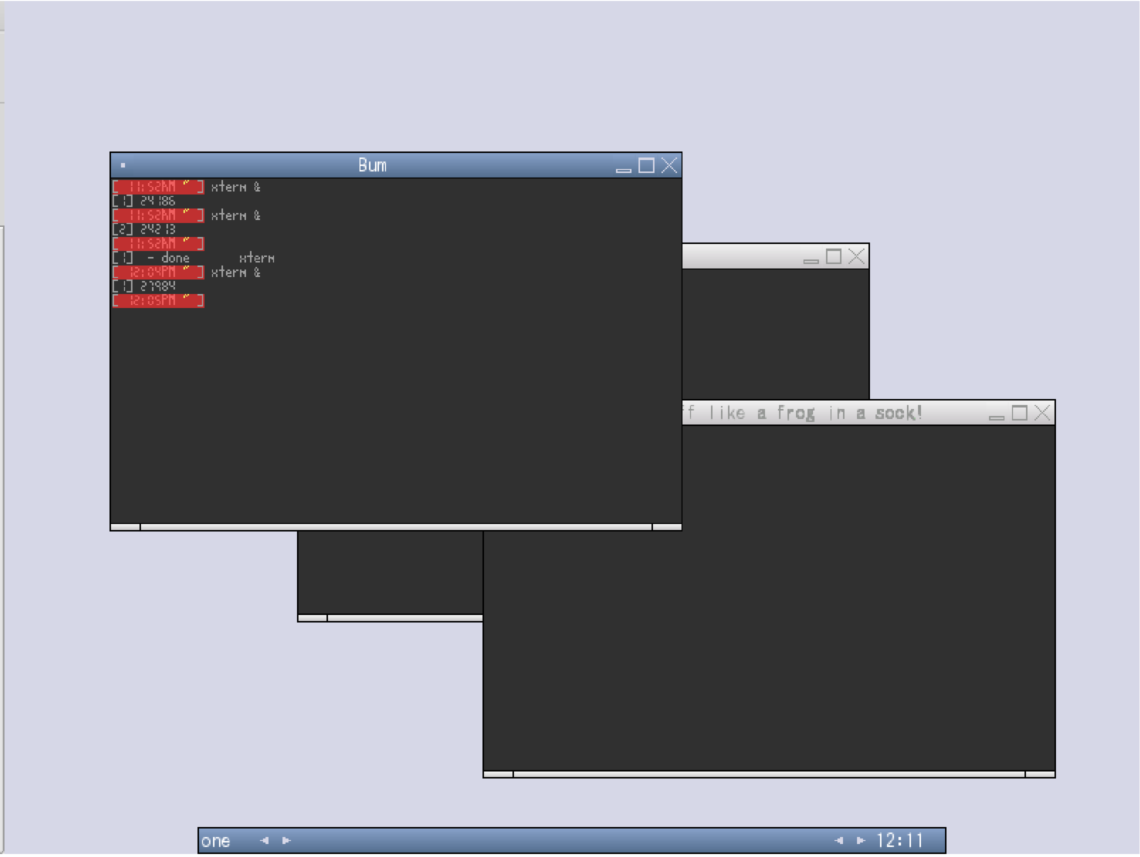
Function	Bytes	Count

```
[ 11:52AM ~ ] xterm &
[1] 24186
[ 11:52AM ~ ] xterm &
[2] 24213
[ 11:52AM ~ ]
[1]  - done       xterm
[ 12:04PM ~ ] xterm &
[1] 27984
[ 12:05PM ~ ] 
```

Fluxbox Memory Use

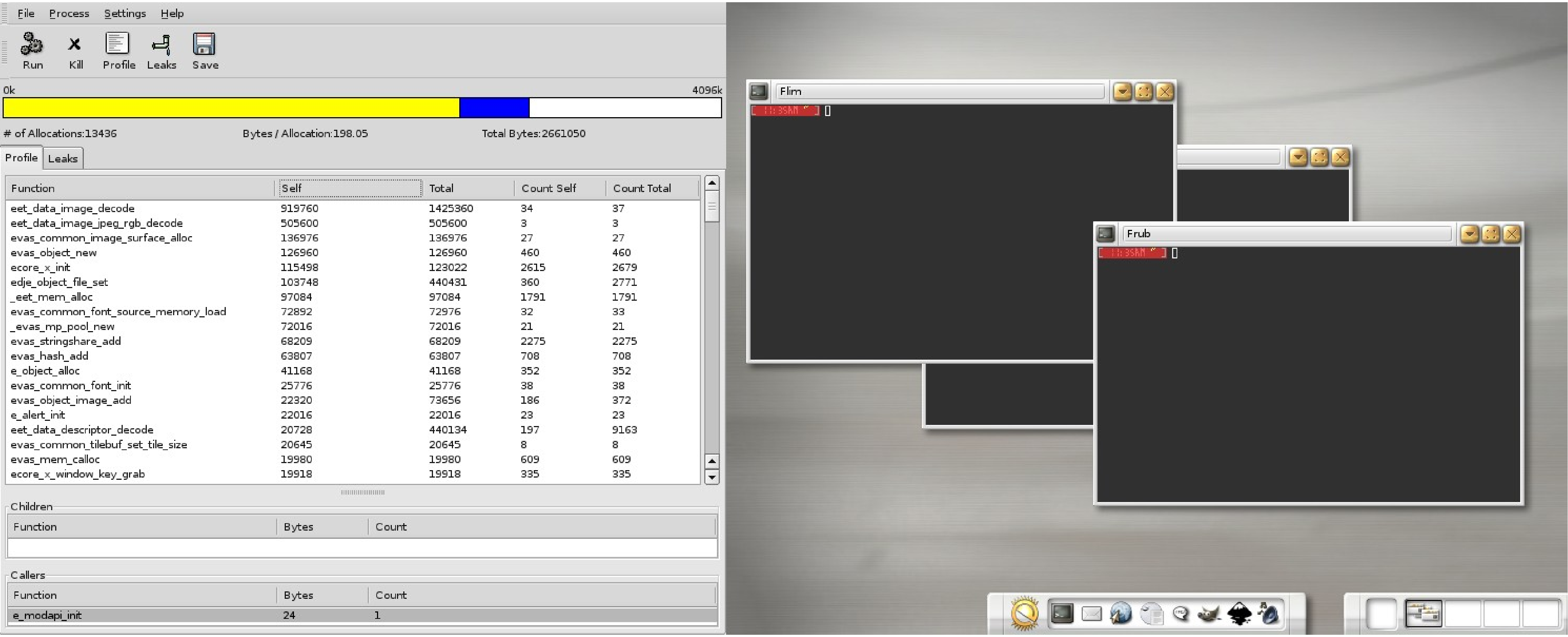


# of Allocations:39742   Bytes / Allocation:59.88   Total Bytes:2379626



```
[ 11:52AM ~ ] xterm &
[1] 24186
[ 11:52AM ~ ] xterm &
[2] 24213
[ 11:52AM ~ ]
[1]  - done       xterm
[ 12:04PM ~ ] xterm &
[1] 27984
[ 12:05PM ~ ]
```

Enlightenment 0.17 Memory Use



# of Allocations:13436 Bytes / Allocation:198.05 Total Bytes:2661050

Function	Self	Total	Count Self	Count Total
eet_data_image_decode	919760	1425360	34	37
eet_data_image_jpeg_rgb_decode	505600	505600	3	3
evas_common_image_surface_alloc	136976	136976	27	27
evas_object_new	126960	126960	460	460
ecore_x_init	115498	123022	2615	2679
edje_object_file_set	103748	440431	360	2771
_eet_mem_alloc	97084	97084	1791	1791
evas_common_font_source_memory_load	72892	72976	32	33
_evas_mp_pool_new	72016	72016	21	21
evas_stringshare_add	68209	68209	2275	2275
evas_hash_add	63807	63807	708	708
e_object_alloc	41168	41168	352	352
evas_common_font_init	25776	25776	38	38
evas_object_image_add	22320	73656	186	372
e_alert_init	22016	22016	23	23
eet_data_descriptor_decode	20728	440134	197	9163
evas_common_tilebuf_set_tile_size	20645	20645	8	8
evas_mem_calloc	19980	19980	609	609
ecore_x_window_key_grab	19918	19918	335	335

Children

Function	Bytes	Count

Callers

Function	Bytes	Count
e_modapi_init	24	1

Did you know...

- Every time a process is executed – it is heavy
- Every X (GTK+/Qt etc.) process needs to connect, make lots of round-trip requests for information and make its own copies of that information
- Different processes tend NOT to share information and tend to duplicate effort
- Add simple features as part of a larger program or as a loadable .so module to save setup costs

Tips

- Don't execute another process unless you REALLY need to
- Share data and resources as it is often expensive to load/decode it multiple times
- Avoid the stampeding herd on startup
- Pre-cache data in memory to avoid waiting on disk IO
- Profile, profile, profile
- Know your code

BLING

Enlightenment 0.17

- Aiming at the minimalist desktop (Desktop Shell)
- Incredibly fast and lean
- Still able to look good with no high-end hardware
- Finishing off all the basics you need to start using your desktop – then release
- Future intentions to be able to use higher end hardware with no loss to those without it

Enlightenment 0.17

- Follow standards
- Attention to detail and optimizations
- Extensible via modules
- Visually highly configurable
- Everything can be animated if desired
- Fast rendering engine (can use software, Xrender, OpenGL and more).
- Multimedia capable

Enlightenment 0.17

- Evas is a state engine
- You only manipulate simple state and don't do expensive drawing most of the time
- Retains state so no need to optimise redraw logic multiple times
- Abstracts the underlying rendering mechanisms allowing for use of a new back-end if/when it becomes feasible

Enlightenment 0.17

- Software (highly optimised)
- Xrender (full support)
- OpenGL (almost full support)
- Framebuffer
- Qtopia
- Others (Cairo, DirectFB)

Enlightenment 0.17

- Xcomposite does NOT make windows transparent
- Xcomposite does NOT provide fancy effects
- It ONLY provides for redirecting window contents from the framebuffer to a pixmap that can then be USED to do the above
- Uses LOTS of video RAM

Enlightenment 0.17

- Xrender can take 2 pixmaps and blend one on the other, rotate and skew images and perform other 2D “advanced” rendering
- Xrender is the “right” API for doing compositing and other advanced 2D tasks
- Has limited rendering quality
- Is mostly unaccelerated and very SLOW
- Is still “the future”

Enlightenment 0.17

- Only open drivers for ATI R200 series chips accelerate Xrender
- No closed drivers accelerate it
- Vendors seemingly not interested in implementing it
- Requires knowledge of advanced chipset features which are kept closed
- Forces us to do “hacks” via OpenGL

Enlightenment 0.17

- Eet (data file storage and compression)
- Evas (2D graphics abstraction)
- Ecore (main loop, events and X, etc. abstractions)
- Embryo (tiny virtual machine engine – much smaller than LUA and much faster than even Java)
- Edje (theme object engine)

Efficiency...

- Buys you the ability to do much more with less
- Allows you to scale DOWN even to embedded devices (100+Mhz ARM etc.)
- Allows those with less $ to enjoy more eyecandy and features
- Shows you care

Enlightenment 0.17

- Pants (Demo time)