Performance <whatever>

Luboš Luňák
<l.lunak@suse.cz>

aKademy 2006

KDE performance

- Is not that good
  - Windows 95 is so much faster
  - And don't let me get started on Jet Pac

- Is not that bad either
  - We are not noticeably worse than comparable competition
  - In fact, we are even often better
  - There's incomparable competition

- So, no need to be very nervous
- But we can still improve

It's bad because

(Some) libraries we use are bad
Dynamic linker (shared libraries) is bad
I/O performance is bad
Really stupid mistakes are bad
Many small things add up
Nice things are sometimes not fast
Unneeded things are done
Initial resource usage is large because our framework is large (libraries)

What to do

- Find the problem
- Analyze the problem
    - Do NOT guess
    - Measure
    - Verify assumptions
    - Speed: cachegrind, sysprof
    - Memory: exmap, xrestop, kmtrace
- Fix the problem

On-demand initialization #1

```
Foo::Foo()
{ ... object = new Object; ... }

Foo::~Foo()
{ ... delete object; }

void Foo::foo()
{ ... object->use(); ... }
```

On-demand initialization #2

```
Foo::Foo()
{ … _object = NULL; … }

Foo::~Foo()
{ … delete _object; }

void Foo::foo()
{ … object()->use(); … }

Object* Foo::object()
{
if( _object == NULL )
      _object = new Object;
return _object;
}
```

Caching

- Don't do the same thing over and over
- Save the result somewhere
- Check that the input hasn't changed
- E.g. ksycoca
- We do many things during startup of every KDE application
    - KConfig, QSettings are rather inefficient
    - Some things are repeated by every application

I/O performance #1

Time for this year's quiz
- 1) loading one 50M file
- 2) loading 1000 x 5k files (=5M)

Which is faster?
- faster is 1)
- it's the same
- faster is 2)

Disk seeks are VERY slow (~10ms)

I/O performance #2

- Try to avoid many small files
- Create a single cache file at build time
- Create a single cache file at runtime
  - Don't forget watching for changes
- On-demand loading
- Kernel could(?) help

Cheat!

Seriously :)
Show progress
Provide early feedback
Optimize the common case
Optimize big bottlenecks

Many small things add up

We are large and complex

"Hundred times nothing tormented the donkey to death" (Slovak proverb)

Time goes on

This KDE will never run on 16MB RAM

We need to compare with comparable competition

We have to live with that

There's a limit that cannot reasonably be reached without significantly reducing features

Would that be still KDE?