

# Vtable Relocations

An approach to reduce unique  
named relocation overhead for  
large C++ application startup.

# The problem:

- Startup is slow:
  - a regular feature in reviews
  - a rationale for 'light-weight' office apps
- Why is startup slow ?

# Overall breakdown:

Self	ELF Object
18.26	libuno_sal.so.3
14.93	configmgr2.uno.so
12.47	ld-2.6.1.so
10.85	libc-2.6.1.so
9.78	libfontconfig.so.1.2.0
5.17	libuno_cppu.so.3
3.88	libexpat.so.1.5.2



# How to speed up linking ?

- Most of the time – we're doing unique named lookup processing:
  - `void dolt();`
  - `void (*global_symbol)() = &dolt;`
- vtables do this -a lot-

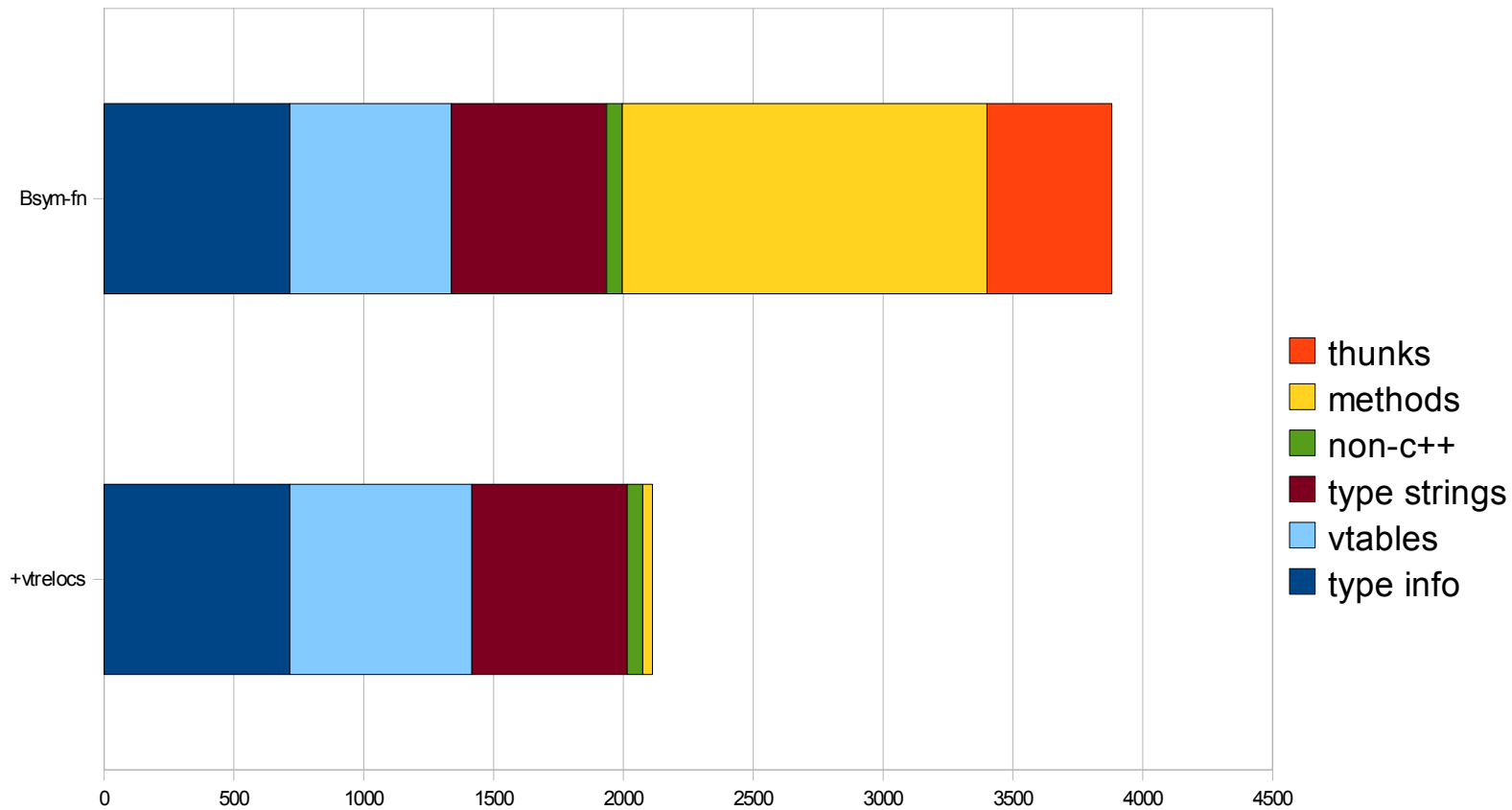


# Vtable relocation tables:

```
_ZVTR_00000006_18SvxXMLXTableImport:  
  .long    _ZTVN3com3sun4star4lang10XUnoTunnelE+8  
  .long    _ZTV18SvxXMLXTableImport+380  
  .long    7  
  .long    _ZTVN3com3sun4star8document7XFilterE+8  
  .long    _ZTV18SvxXMLXTableImport+352  
  .long    7  
  .long    _ZTVN3com3sun4star8document9XImporterE+8  
  .long    _ZTV18SvxXMLXTableImport+328  
  .long    7  
  .long    _ZTVN3com3sun4star4lang15XInitializationE+8  
  .long    _ZTV18SvxXMLXTableImport+304  
  .long    7  
  .long    _ZTVN3com3sun4star4lang12XServiceInfoE+8  
  .long    _ZTV18SvxXMLXTableImport+272  
  .long    7  
  .long    _ZTVN3com3sun4star3xml3sax24XExtendedDocumentHandlerE+8  
  .long    _ZTV18SvxXMLXTableImport+200  
  .long    2047  
  .long    _ZTVN3com3sun4star4lang13XTypeProviderE+8  
  .long    _ZTV18SvxXMLXTableImport+172  
  .long    7  
  .long    _ZTV11SvXMLImport+136  
  .long    _ZTV18SvxXMLXTableImport+136  
  .long    127  
  .long    _ZTV11SvXMLImport+8  
  .long    _ZTV18SvxXMLXTableImport+8  
  .long    -305
```

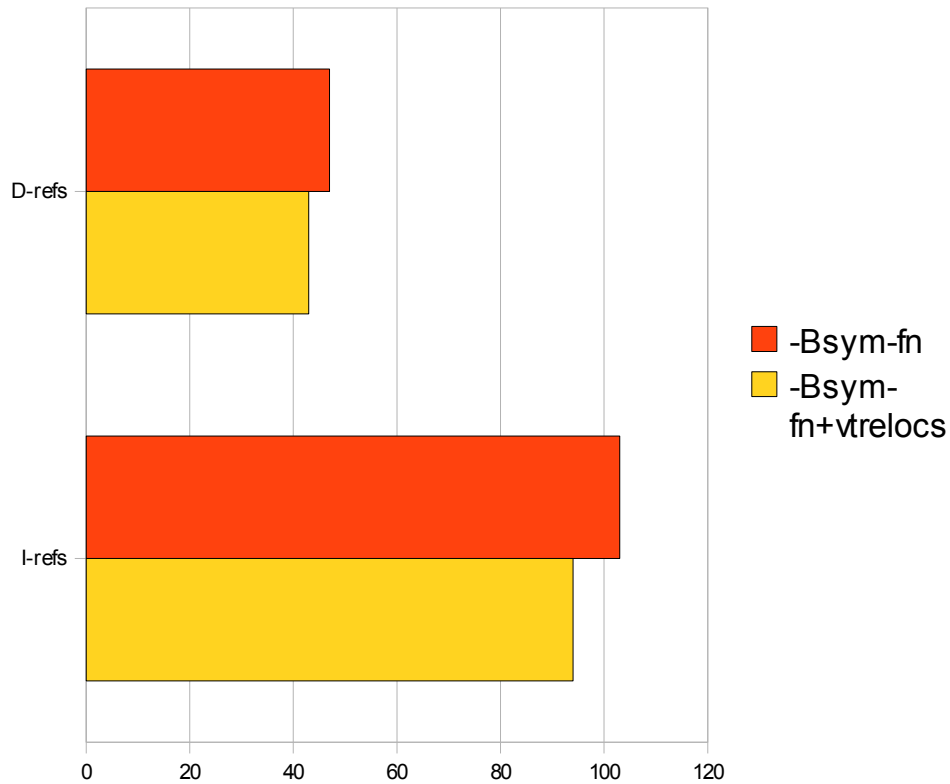
# Savings for libsvx

## Unique Named Relocations:

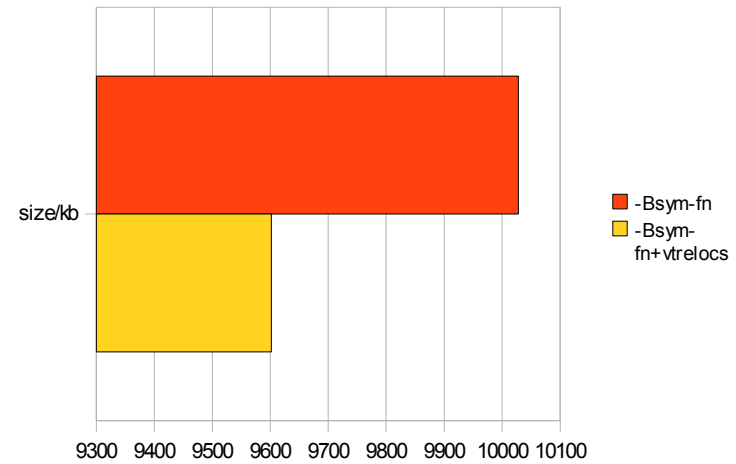


# Savings for libsvx (2)

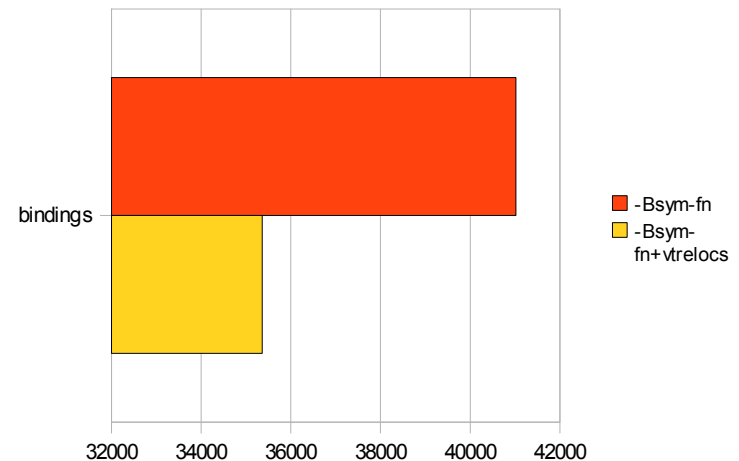
dlopen time: millions of cycles.



Size savings: ~5%



Binding savings (writer startup)



empirically: `$ readelf -r -W libsvx680li.stock | cut -c '54-' | uniq | less`

<code>__cxa_pure_virtual</code>	<code>_ZTV10SdrObjList</code>
<code>_ZTVN10__cxxabiv120__si_class_type_infoE</code>	<code>_ZTV6SdrHdl</code>
<code>_ZTVN10__cxxabiv117__class_type_infoE</code>	<code>_ZTV11SfxPoolItem</code>
<code>_ZTI5Timer</code>	<code>_ZTV100faPtrItem</code>
<code>_ZNK3sdr7overlay130overlayObject5isHitERKN7basegfx8B2DPointE</code>	<code>_ZTV10SvxBoxItem</code>
<code>_ZN3sdr7overlay290overlayObjectWithBasePosition9transformERKN7basegfx12B2DH</code>	<code>_ZTV10SvxGrfCrop</code>
<code>omMatrixE</code>	<code>_ZTV7SfxHint</code>
<code>_ZN3sdr7overlay130overlayObject26stripeDefinitionHasChangedEv</code>	<code>_ZTVN3vcl9unohelper17D</code>
<code>_ZTIN3sdr7overlay290overlayObjectWithBasePositionE</code>	<code>ragAndDropClientE</code>
<code>_ZN3sdr7overlay130overlayObject7TriggerEm</code>	<code>_ZTV13SdrDragMethod</code>
<code>_ZTIN3sdr7overlay150overlayBitmapExE</code>	<code>_ZTV11SdrDragMove</code>
<code>_ZN3sdr7overlay150overlayBitmapExD1Ev</code>	<code>_ZTV11SdrHdlColor</code>
<code>_ZN3sdr7overlay150overlayBitmapExD0Ev</code>	<code>_ZTV8Outliner</code>
<code>_ZN3sdr7overlay150overlayBitmapEx12drawGeometryER12OutputDevice</code>	<code>_ZTV11SdrOutliner</code>
<code>_ZN3sdr7overlay150overlayBitmapEx15createBaseRangeER12OutputDevice</code>	<code>_ZTV11SfxListener</code>
<code>_ZN3sdr7overlay150overlayBitmapEx14zoomHasChangedEv</code>	<code>_ZTV14SdrObjUserData</code>
<code>_ZTSN3sdr7overlay150overlayBitmapExE</code>	<code>_ZTV11SvxFontItem</code>
<code>_ZN3sdr7overlay130overlayObject14zoomHasChangedEv</code>	<code>_ZTV11SvxGridItem</code>
<code>_ZTIN3sdr7overlay140overlayManagerE</code>	<code>_ZTV11SvxLineItem</code>
<code>_ZN3sdr7overlay140overlayManagerD1Ev</code>	<code>_ZTV11SvxPageItem</code>
<code>_ZN3sdr7overlay140overlayManagerD0Ev</code>	<code>_ZTV11SvxSizeItem</code>
<code>_ZNK3sdr7overlay140overlayManager14completeRedrawERK6RegionP12OutputDevice</code>	<code>_ZTV14SfxChildWindow</code>
<code>_ZN3sdr7overlay140overlayManager5flushEv</code>	<code>_ZTV14SfxBroadcaster</code>
<code>_ZN3sdr7overlay140overlayManager8copyAreaERK5PointS4_RK4Size</code>	<code>_ZTV11OCX_Control</code>
<code>_ZNK3sdr7overlay140overlayManager17restoreBackgroundERK6Region</code>	<code>_ZTV12SdrPaintView</code>
<code>_ZN3sdr7overlay140overlayManager15invalidateRangeERKN7basegfx8B2DRangeE</code>	<code>_ZTV12SvxBrushItem</code>
<code>_ZTSN3sdr7overlay140overlayManagerE</code>	<code>_ZTV12SvxColorItem</code>
<code>_ZN3sdr7overlay290overlayObjectWithBasePositionD1Ev</code>	<code>_ZTV12SvxFieldItem</code>
<code>_ZN3sdr7overlay290overlayObjectWithBasePositionD0Ev</code>	<code>_ZTV17SfxControllerItem</code>
<code>_ZTIN3sdr7overlay130overlayObjectE</code>	<code>m</code>
<code>_ZN3sdr7overlay130overlayObjectD1Ev</code>	<code>_ZTV13SdrDragObjOwn</code>
<code>_ZN3sdr7overlay130overlayObjectD0Ev</code>	<code>_ZTV13SdrDragResize</code>
<code>_ZTSN3sdr7overlay290overlayObjectWithBasePositionE</code>	<code>_ZTV13SfxUndoAction</code>
<code>_ZTSN3sdr7overlay130overlayObjectE</code>	<code>_ZTV13SdrUndoAction</code>
<code>_ZN3sdr7overlay250overlayPolyPolygonStriped15createBaseRangeER12OutputDevice</code>	<code>_ZTV13SvxBulletItem</code>
<code>e</code>	<code>_ZTV13SvxColumnItem</code>
<code>_ZNK3sdr7overlay250overlayPolyPolygonStriped5isHitERKN7basegfx8B2DPointE</code>	<code>_ZTV13SvxDoubleItem</code>
	<code>_ZTV13SvxMarginItem</code>



# Conclusion:

- A set of useful wins
- More space/time saving work is possible
  - breaking the C++ ABI as a side-effect
    - private thunks, direct vtable calls etc.
  - existing DSO size breakdown:

