
OpenOffice.org
Conference 2008 Beijing
世界开源大会
CREATED WITH LOTUS® SYMPHONY™

OpenOffice.org Conference 2008 Beijing

Symphony performance improvement best practices

Cheng Yuan, Wang Lei

Symphony performance team, IBM

IBM Lotus® Symphony™ Overview

- A set of office productivity applications
  - Create, edit, share documents, spreadsheets, and presentations
  - Based on OpenOffice.org and Eclipse
  - Support OpenDocument Format (ODF)
  - Support Microsoft® Office and other formats
- Available at *no charge* to anyone and everyone
  - www.ibm.com/software/lotus/symphony
  - Web-based support and community forums

Agenda

- **Performance Methodology**
- **Symphony Performance Benchmark**
- **Performance improvements in Symphony**
  - Startup performance improvement
  - Asynchronous loading
  - Partial saving
- **Q&A**

Symphony Performance Methodology (1)

- **Client performance definition :**
  - Response time is the key. (Memory footprint eventually impact response time)
- **Analyzing methodology:**
  - Firstly classify the problem : IO-bound vs Compute-intensive.
    - Cold run, warm run, using ramdisk tool.
  - Find bottleneck at coding level :
    - Monitor tools: Perfmon(all), filemon(IO), processexplore(all), sysstat(all), iostat(IO),vmstat(memory)
    - Instrument/profiling tools : jprofiler(java instrument), IBM rational quantify(c++ instrument), valgrind (c++ instrument & memory), IBM rational purify(memory), WinDbg(stack profiling), gdb (stack profiling), dependency walker(startup profiling), ldd(startup profiling)
  - Find redesign opportunity at design level:
    - Study the design in Symphony from performance perspective.
    - Familiar with infrastructure / framework / application level of Symphony.

Symphony Performance Methodology (2)

- **Optimizing methodology:**
  - IO-bound bottleneck :
    - Prefetch files, merge small files, optimize DLL loading, code reposition, cache to reduce IO, overlap IO and CPU time.
  - Compute-intensive bottleneck:
    - Remove redundant code, string optimization, reduce memory alloc/free and mutex, defer on-demand work, minimize inter-process calls, using high-performance library.
  - Language special optimization:
    - Java (JVM tuning[shareclass, notrace...], delay activating bundles ...)
    - C++ (compiler options, using fix-mem-pool, inline funciton, reference ...)
  - No bottleneck, need redesign:
    - Preload, partial saving, picture on-demand loading, spell-check redesign.
    - Perceived Performance : threading(e.g. Not block UI when doing long-playing work), streaming (e.g. Asynchronous loading)

Agenda

- **Performance Methodology**
- **Symphony Performance Benchmark**
- **Performance improvements in Symphony**
  - Startup performance improvement
  - Asynchronous loading
  - Partial saving
- **Q&A**

Performance benchmark(1)

OpenOffice.org Conference 2008 Beijing

Benchmark = Sample files + User Profile

Name	Content	Size(KB)	Filter
sw_complex_100p.doc	100pages,100pictures,16tables,60textboxes,	3,082,752	doc
sw_complex_100p.odt	100pages,100pictures,16tables,60textboxes,	1,249,078	odt
sw_complex_100p.sxw	100pages,100pictures,16tables,60textboxes,	1,246,190	sxw
sw_plain_120p.doc	120pages,text and bulletin	672,768	doc
sw_plain_120p.odt	120pages,text and bulletin	19,331	odt
sw_plain_120p.sxw	120pages,text and bulletin	13,105	sxw
sd_complex_51p.odp	51pages,1700objects	1,236,564	odp
sd_complex_51p.ppt	51pages,1700objects	1,812,992	ppt
sd_complex_51p.sxi	51pages,1700objects	1,224,475	sxi
sd_plain_50p.odp	50pages,170textboxes, pure text	26,943	odp
sd_plain_50p.ppt	50pages,170textboxes, pure text	140,288	ppt
sd_plain_50p.sxi	50pages,170textboxes, pure text	26,785	sxi
sc_complex_13sh_4kcell.ods	13sheets,4k cells with format	139,411	ods
sc_complex_13sh_4kcell.sxc	13sheets,4k cells with format	117,835	sxc
sc_complex_13sh_4kcell.xls	13sheets,4k cells with format	281,600	xls
sc_plain_4sh_5kcell.ods	4sheets,5k cells	110,161	ods
sc_plain_4sh_5kcell.sxc	4sheets,5k cells	78,063	sxc
sc_plain_4sh_5kcell.xls	4sheets,5k cells	159,232	xls

Performance benchmark(2)

- **Benchmark Priority**

  $(T_{Symphony} - T_{MS\ Office})$ x **Frequency** x **Importance**

  - **Frequency**: How often the action will be done by a user per day.
  - **Importance**: e.g. Save ODF > Load ODF > Load/Save MS Office Formats

    e.g. Load show(1st page show) > Load finish(all pages completely loaded)

- **Benchmark categories**
  - Startup and Shutdown, File Operations(load/save), Symphony Operations
- **Benchmark automation**
  - Using IBM RFT(Rational Functional Tester).
  - Automated startup, loading, saving cases.

Agenda

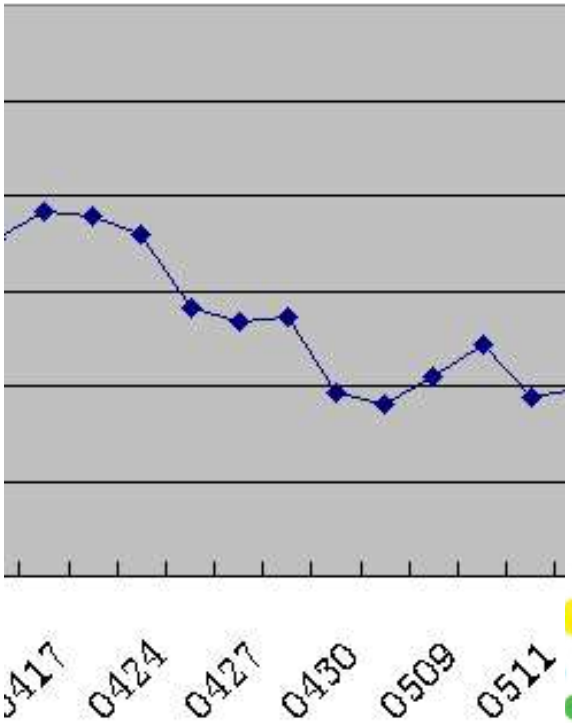
- **Performance Methodology**
- **Symphony Performance Benchmark**
- **Performance improvements in Symphony**
  - Startup performance improvement
  - Asynchronous loading
  - Partial saving
- **Q&A**

Practice 1 - Startup Performance

Startup Performance

- **Cold and warm startup**
  - Cold startup : Restart the computer, then startup the application (the 1st startup)
  - Warm startup : Just startup the application again after the 1st startup
- **Automated test of startup performance**
  - Create one empty NTFS partition to install Symphony.
  - Put the 'Start.bat' in OS startup folder:

```
cd "C:\Documents and Settings\tester"
sleeptool.exe 200
call javalauncher.bat
sleeptool.exe 30
shutdown -r -t 10
```



Startup Performance tuning

1. **Merge small libraries into bigger one to reduce the library-loading overhead. 14% improvement**
2. **Remove some unused code in Symphony startup path to reduce the computation workload. 3% improvement.**
3. **Optimize non-library file IO, prefetch rdb files. 5% improvement.**
4. **Reduce export symbols of library to try to reduce I/O. No improvement.**
5. **Use OS tools such as rebase. Downgrade 15% .**

   Theoretically rebased applications can reduce the 'relocation' step and be supposed to improve the startup performance. But actually cold startup procedure will incur many hard page fault, including much disk-seeking time. For not-rebased application, OS will read the DLLs into memory sequentially for relocation, these sequential read will eliminate most disk-seeking time.

Practice 2 - Asynchronous loading

What is Asynchronous Loading

- **Synchronous loading :**
  - The loading process of Symphony is consisted of two steps:
    - Load (filter): Read the document data from the disk and construct the content model in the memory.
    - CreateView: Construct the view model according to the content model and display to the users.
  - In previous version of Symphony, the two steps are executed in sequence, so the users have to wait until the entire document data have been loaded into memory and formatted to display.
- **Asynchronous loading in Symphony:**
  - To improve the loading performance, Symphony make the two steps executed in two threads, so the user can view/edit part of the document content during the document is loading.

Asynchronous loading design

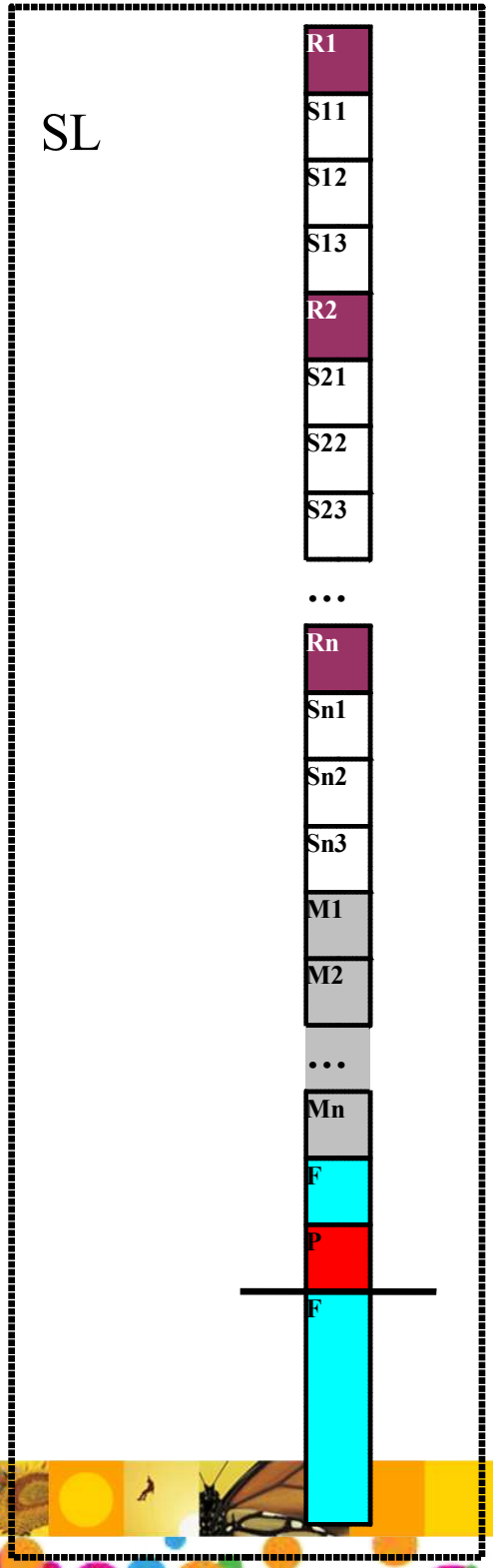
- **Register the AL enable filters by filter names.**
  - odt, odp, doc, ppt, lwp, prz
- **Create a new load thread**
  - Execute loading in a separate thread(the load thread), and the main thread can continue to execute the CreateView() to display the content.
  - Load thread : load (read,setattr)
  - Main thread : makefrm,format,paint
- **Interaction between loadthread and mainthread**
  - *LoadFinish*: (set by load thread) *ReadyToShow*: (set by load thread)*StopFlag*: (set by main thread) *StartReadElements*: (set by main thread)

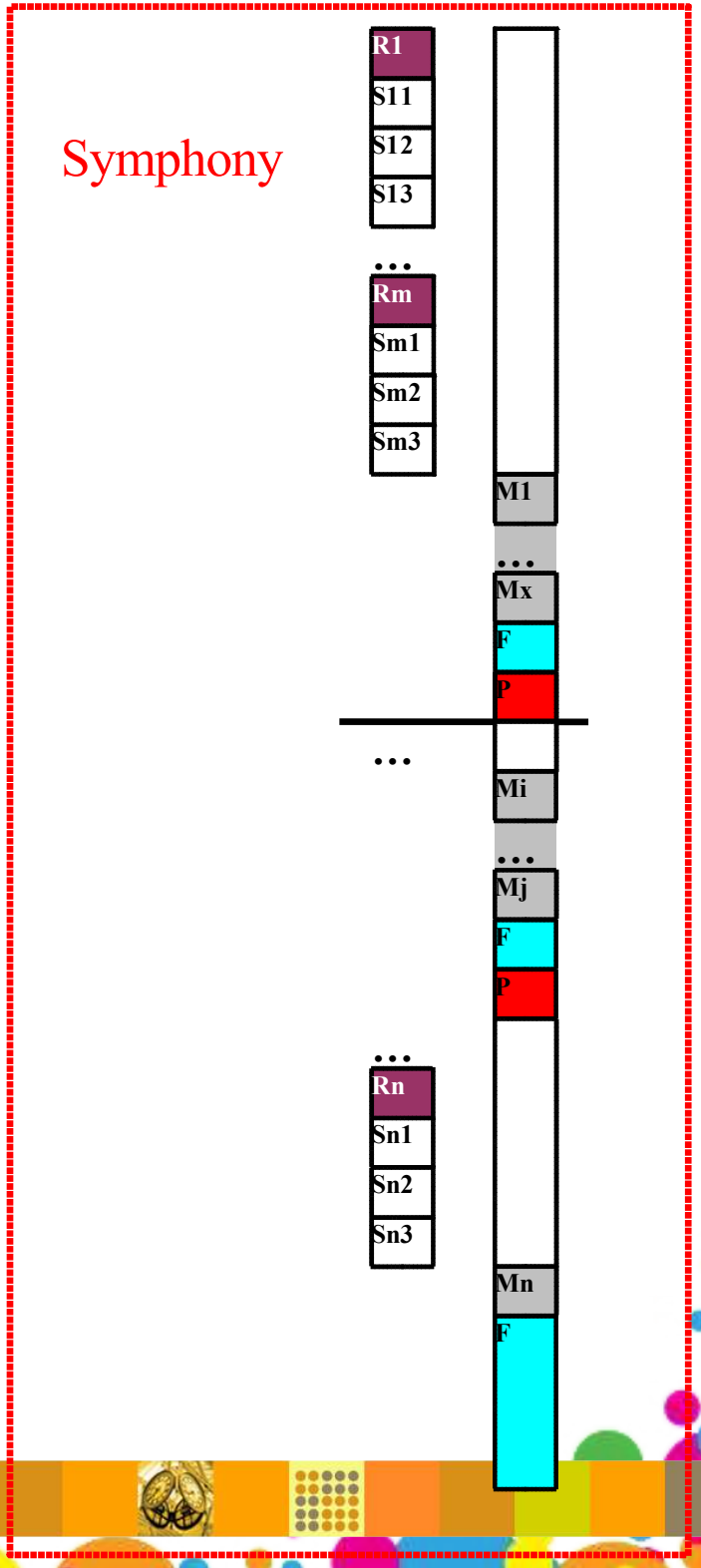
Asynchronous loading for Document

Read
Setattr
MakeFrame
Format
Paint

The right time to format Only format these nodes: all the attrs have been set .

The right time to paint Paint only the validate and ‘fulfilled’ page.





Asynchronous loading for Presentation

- **Relative simple : no formatting process**
- **The tree model : sdrmodel, broadcast mechanism**
- **Model change will notify the 4 view : page view, outliner view, notes view, page sorter view.**
- **Hold on 1 page as buffer: when reading n page, just show before n-1 pages.**

Support different filters

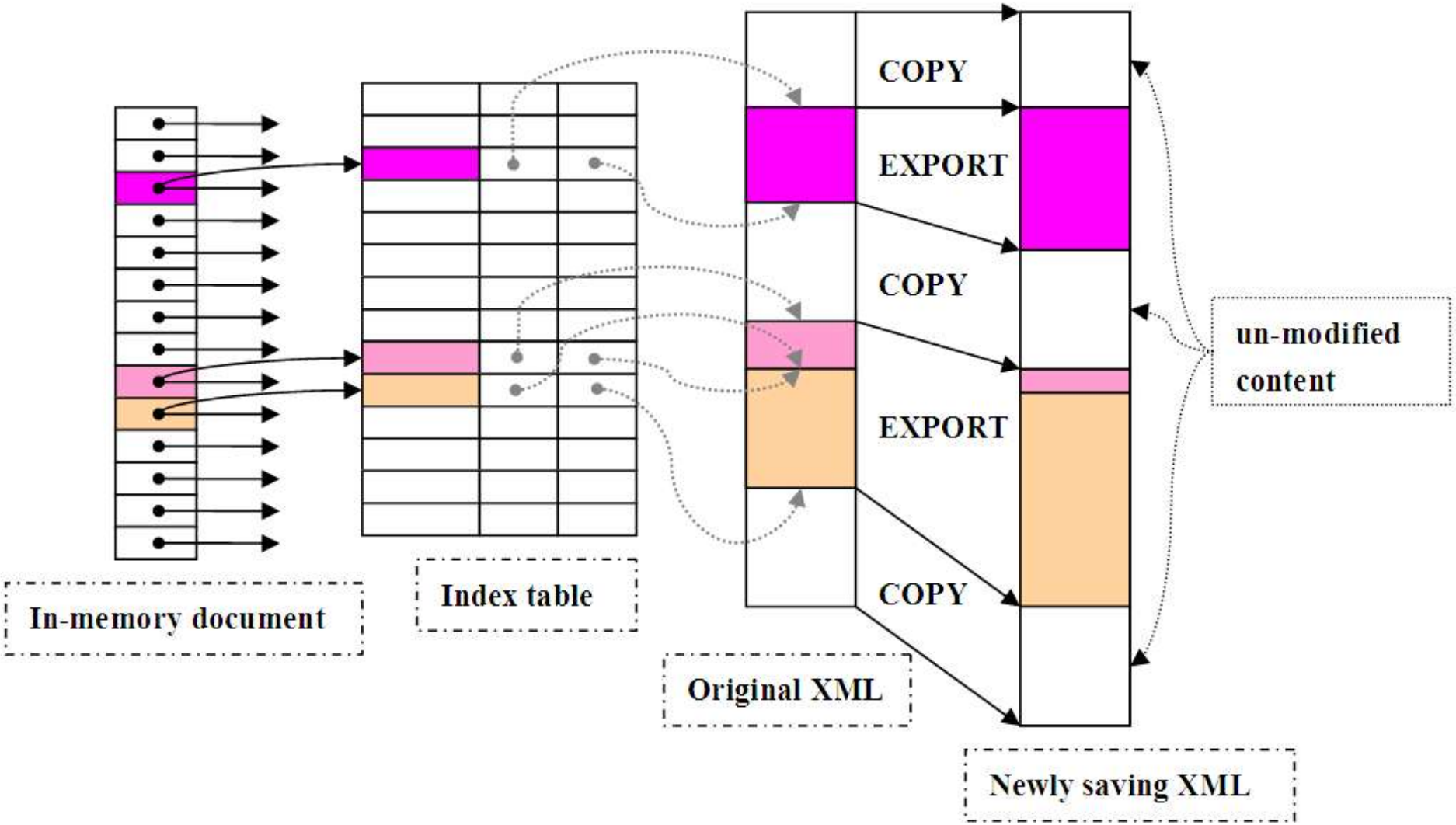
1. Try to filter in the ‘right’ sequence.
2. Filter must NOT touch the view model.
3. It’s suggested that filter uses the UNO interface of the content model when constructing the content model.
4. Filter must use mutex to protect the share variables between loading thread and main thread. (or SolarMutex)
5. May need to call the Application::Reschedule() in proper time in the time-consume operations in filter.

Practice 3 - Partial saving

What is partial saving

- **Normal saving :**
  - In previous version, when saving Symphony will export the whole document into one new file from scratch even if user just change one character.
  - Common user mostly prefer to save frequently when editing. For large document including complex objects and pictures, the saving procedure is always slow, which will interrupt user's editing process.
- **Partial saving in Symphony:**
  - Only export the modified parts. Copy unchanged xml fragments, export changed parts. (Copying is much faster than exporting)
  - Using page(ODP) or paragraph(ODT) as a saving unit.

Partial saving design



Key points about ODP partial saving

- **Index table**
  - Record the start and end offset of an XML block.
- **Page changed flag**
  - Indicate whether a presentation page is modified by user.
- **Sub streams name repository**
  - Store the names of child streams. Add/remove the name during saving. Copy the child stream directly if unchanged.
- **Deleted page list**
  - Maintain the list of deleted pages. If a presentation page is deleted, the partial saving process should skip copying the bytes of deleted page from source storage.

Agenda

- **Performance Methodology**
- **Symphony Performance Benchmark**
- **Performance improvements in Symphony**
  - Startup performance improvement
  - Asynchronous loading
  - Partial saving
- **Q&A**

Discussion of performance

- **Known performance problems?**
  - e.g. Startup, loading, saving, painting.
- **Performance concern of xml file format?**
  - Small part xml, Index and relationship

Reference: Symphony Sessions

- IBM Lotus Symphony Technical Overview
- Lotus Symphony extension model
- Accessibility in Symphony
- Symphony performance improvement best practices
- Introduction of SMIL Animation and Implementation in IBM Lotus Symphony
- Properties Sidebar, make editing much easier
- Learn more about office users - Feature usage study by document element statistics
- Visualize Writer Document Structure for Productive Development

OpenOffice.org
Conference 2008 Beijing

OpenOffice.org
Conference 2008 Beijing

Thanks!

凝聚全球力量 绽放开源梦想

www.OOobeijing2008.com