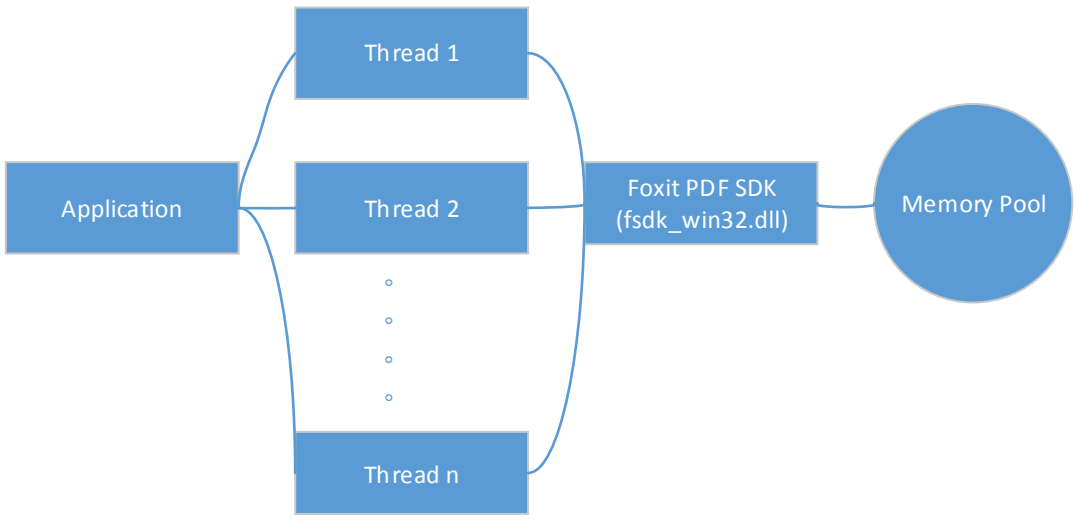
How to improve performance of applications based on Foxit PDF SDK in servers ?

Nowadays, most servers have excellent hardware performance with multi-core CPU and large capacity memory. Foxit has proposed a solution to utilize these hardware advantages for machines with multi-core CPU. This solution has been issued as a Chinese Patent (Patent Number: ZL 2010 1 0140502.6). And it's also in the process of applying for a US Patent through PCT, a patent agency in U.S. Currently, it's in the phase of publication (Publication Number: US 20130061018 A1).

Below are technical introduction and implementation of this solution:

Based on plenty of experiments, we find that operations of PDF, especially parsing process, need to access memory frequently. Usually, multiple threads share the same memory pool in a process, such as Foxit PDF SDK. So thread safety should be ensured for memory access. In this way, a lot of time is exhausted when multiple threads want to access memory synchronously, which becomes a bottleneck. What's worse, this bottleneck causes multiple threads can't be dispatched to CPU in time. That means CPU resources can't be utilized adequately, neither can hardware performance of server. Finally, performance of applications will be restricted in a large extent.

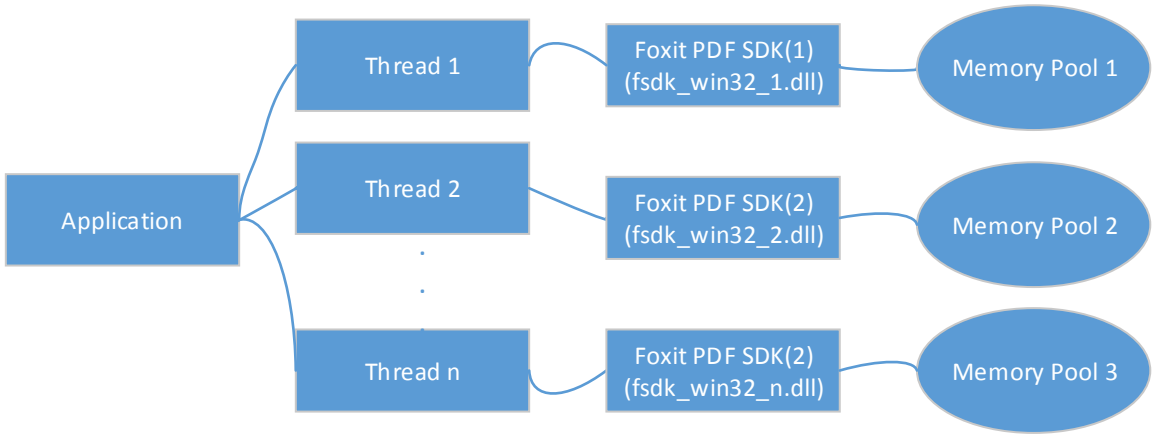
Schematic:



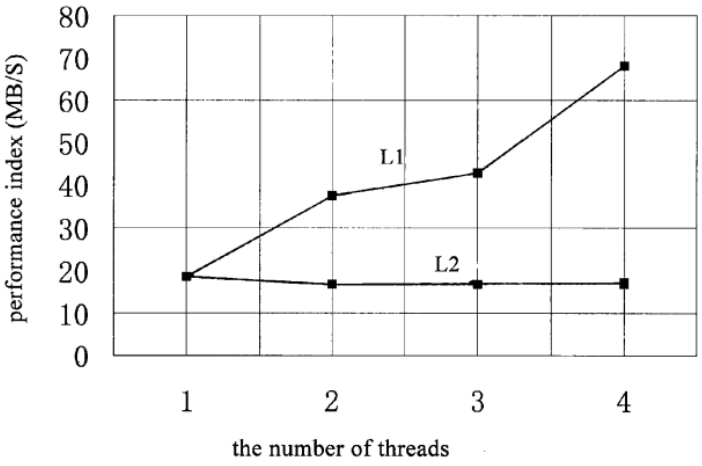
As shown above, all threads will be dispatched by Foxit PDF SDK to access the same memory pool, and it causes the downgrade in efficiency of synchronization. So frequent memory access becomes the bottleneck of performance enhancement.

For Windows server, we raise a proposal to resolve this bottleneck: let each thread have its own memory space by making copy of process for each. The approach is applied in Foxit PDF SDK by making each thread have a copy of Foxit PDF SDK dynamic library. In this way, each thread can access its own

memory rather than the same memory pool. It eliminates the waiting time in threads synchronization and enhance performance of applications. Schematic is shown as below:



The result from experiments can prove that in our solution, performance of applications will be enhanced linearly with the increment of CPU number. The following diagram shows the relationship between performance and the number of threads. In this diagram, L1 represents performance adopting our solution which means to make copies of SDK library for each thread, while L2 indicates performance without our solution which means all thread share the same memory pool.



From this diagram, we can clearly see performance of L1 is enhanced from 20MB/s to about 70MB/s when the number of threads increases from 1 to 4. As a reference, performance of L2 keeps almost the same when the number of threads increases.

In order to help developers to understand how to use this approach, Foxit also provides a demo named “mt_PDF2txt” in directory “samples” of the downloaded package for Windows, to introduce how to implement multi-thread applications based on Foxit PDF SDK to achieve higher performance. This demo performs the conversion from PDF to text with multi-thread support.

For Linux server, memory allocation mechanism of Linux is different from Windows server. So this approach cannot be used directly in Linux sever. To resolve this bottleneck in Linux server, application

should use multi-process method. Foxit provides a demo named “linux_mt_pdf2txt” in directory “samples” of the downloaded package for Linux, to introduce how to implement multi-process applications based on Foxit PDF SDK to achieve higher performance. This demo performs the conversion from PDF to text with multi-process support.

In addition, this approach has been applied in Foxit PDF IFilter, which is one of Foxit product and mainly targets in servers and provides quick searching service to applications or users. For more detailed introduction about this product, please visit the following website:

http://www.foxitsoftware.com/products/ifilter/performance.php

Below are reviews from MSDN about the performance enhancement of Foxit PDF IFilter.

http://blogs.msdn.com/b/opal/archive/2010/02/09/pdf-ifilter-test-with-sharepoint-2010.aspx