


## Improving performance (stc)

[Classic](#) [List](#) [Threaded](#)

11 messages [Options](#)

Apr 18, 2009; 01:55am  [eishay](#) - Improving performance (stc) [Reply](#) [More](#) [CLOSE](#)

Hi,

Please let me know if there is a better place for such discussion. I noticed that we can make the loop that does the create links in base clusters graph at STCEngine.createMergedClusters run in threads and by that make it much faster for large amount of documents. Using it on 8k documents it decreased the time for the same code section from 67971ms to 20465ms (more then three times faster).

Here is an example code, please let me know if there is a better way to share it:


```
private void createLinksInBaseClustersGraph(final float MERGE_THRESHOLD)
{
    //I use a four cores machine.
    ThreadPoolExecutor tpe = new ThreadPoolExecutor(4, 4, 1000, TimeUnit.SECONDS, new ArrayBlockingQueue<Runnable>(baseClusters.size()));
    List<FutureTask> list = new ArrayList<FutureTask>();
    for (int i = 1; i < baseClusters.size(); i++)
    {
        final int index = i;
        FutureTask task = new FutureTask(new Callable(){

            public Object call() throws Exception
            {
                BaseCluster a = (BaseCluster) baseClusters.elementAt(index);
                final long a_docCount = a.getNode().getSuffixedDocumentsCount();

                for (int j = 0; j < index; j++)
                {
                    BaseCluster b = (BaseCluster) baseClusters.elementAt(j);

                    final double a_and_b_docCount = a.getNode()
                        .getInternalDocumentsRepresentation().numberOfSetBitsAfterAnd(
                            b.getNode().getInternalDocumentsRepresentation());

                    // BUG: This check should be bidirectional (see Zamir's paper).
                    if (((a_and_b_docCount / b.getNode().getSuffixedDocumentsCount()) > MERGE_THRESHOLD)
                        && ((a_and_b_docCount / a_docCount) > MERGE_THRESHOLD))
                    {
                        // add links to base cluster graph. This is actually redundant as
                        // we're adding two
                        // directed edges.
                        a.addLink(b);
                        b.addLink(a);
                    }
                }
            }
        });
        return null;
    });
    tpe.execute(task);
    list.add(task);
}
for(FutureTask task: list)
{
    try
    {
        task.get();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

- Apr 18 [JIRA dawid.weiss@cs.put.poznan.pl](#) Hi Eishay, This patch looks sensible -- I guess it will speed up things on multi-core processors, but...
  - Apr 18 [eishay](#) Absolutely Dawid, the snippet wasn't meant to be a patch :-) I actually thought about some generalization of the...
  - Apr 18 [JIRA dawid.weiss@cs.put.poznan.pl](#) > I actually thought about some generalization of the pool. There might be more > places in the code where we could...
- Apr 19, 2009; 05:35am  [eishay](#) - Re: Improving performance (stc) [Reply](#) [More](#) [CLOSE](#)

> I do get your point. We generally consider a clustering process to be  
 > single-threaded (because in search results clustering scenario processing cores  
 > are saturated by HTTP requests anyway). Multi-threading is for running multiple  
 > clustering processes concurrently and this can be done safely via  
 > CachingController, for example.  
 I'll check the use of CachingController for throughput, but right now I'm more concerned about the latency. Though I know I cannot make it fast enough for online processing with the data profile I'm going to use, I would like to have it complete asap.

> I guess what I'm trying to say is that I don't have any good hints as to where  
 > such a generalization could be placed in the existing codebase. Since you're  
 > working with larger data sets it's your call to suggest something -- if anybody  
 > doesn't like it, you can count on our assertiveness :)  
 I'm examining the application with a profiler (YourKit) and trying find parallelize only in places it is possible and really takes a significant amount of cpu time. I'll keep you updated :-)

>> Can you give me a pointer to this? Sounds interesting. Java6 is really a huge  
 >> boost in performance. We will try to keep backward compatibility with 1.5, but I  
 >> would tell everyone to make the switch to 1.6 asap.  
 FJ is really cool:  
<http://www.infoq.com/news/2007/07/concurrency-java-se-7>  
<http://gee.cs.oswego.edu/dl/papers/fj.pdf>  
 Unfortunately I'll be stuck in the next few months with java5 on my development environment (waiting for Apple to fix a bug which is a blocker for me). But as soon I'll be able to use java6 I'll convert whatever threadpool code I'll do to FJ.

<Scala fanboy mumbling to himself> Another option is to use Scala's Actors which is also based on FJ and runs on java5 :-) <end of mumbling>

Eishay
- Apr 20 [JIRA dawid.weiss@cs.put.poznan.pl](#) Interesting reading, thanks. I don't think switching to FJ can improve things a lot in our case because the number...
  - Apr 20 [Stanislaw Osinski](#) > > Interesting reading, thanks. I don't think switching to FJ can improve > things a > lot in our case because the...
  - Apr 21 [eishay](#) Thanks Stanislaw, I'll look into it. Eishay
  - Apr 18 [zhu hui](#) Hi, all I am a new comer to carrot, and nowadays want to build a academic paper search engine with clustering...
  - Apr 18 [eishay](#) Hi Eric, I totally makes sense. I am trying to use Carrot in a bit different way, i.e. not for real time but more of...
  - Apr 19 [zhu hui](#) Hi, Eishay Thanks very much for your kindly reply. I have just readed the doctor thesis of David Weiss. I...

« [Return to forum](#) | 34 views