Java fork/join implementation

Jeremy Singer

University of Glasgow

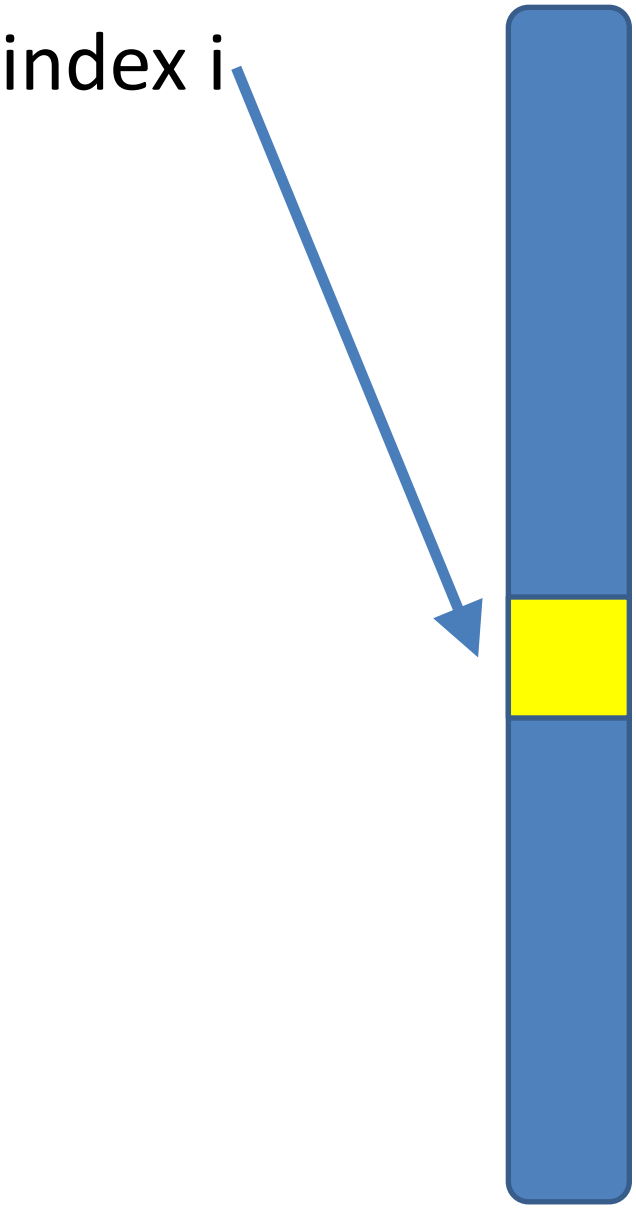
```
ConcurrentHashMap
  <String, Vector<Integer>>
```

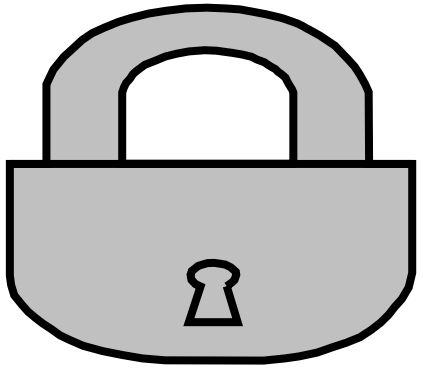
String is key : "Zadok the priest"

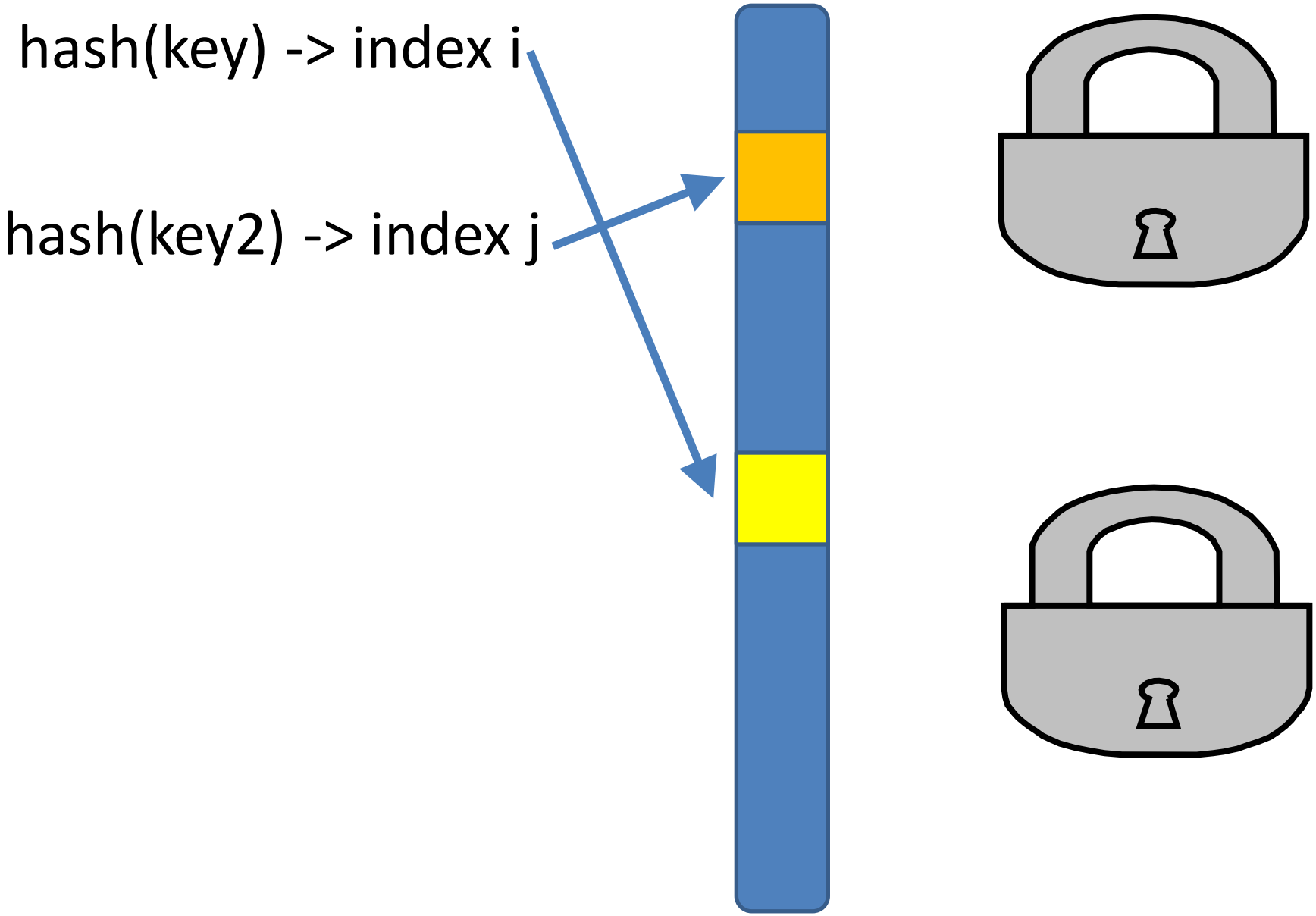
Integer is word position in file: 23641

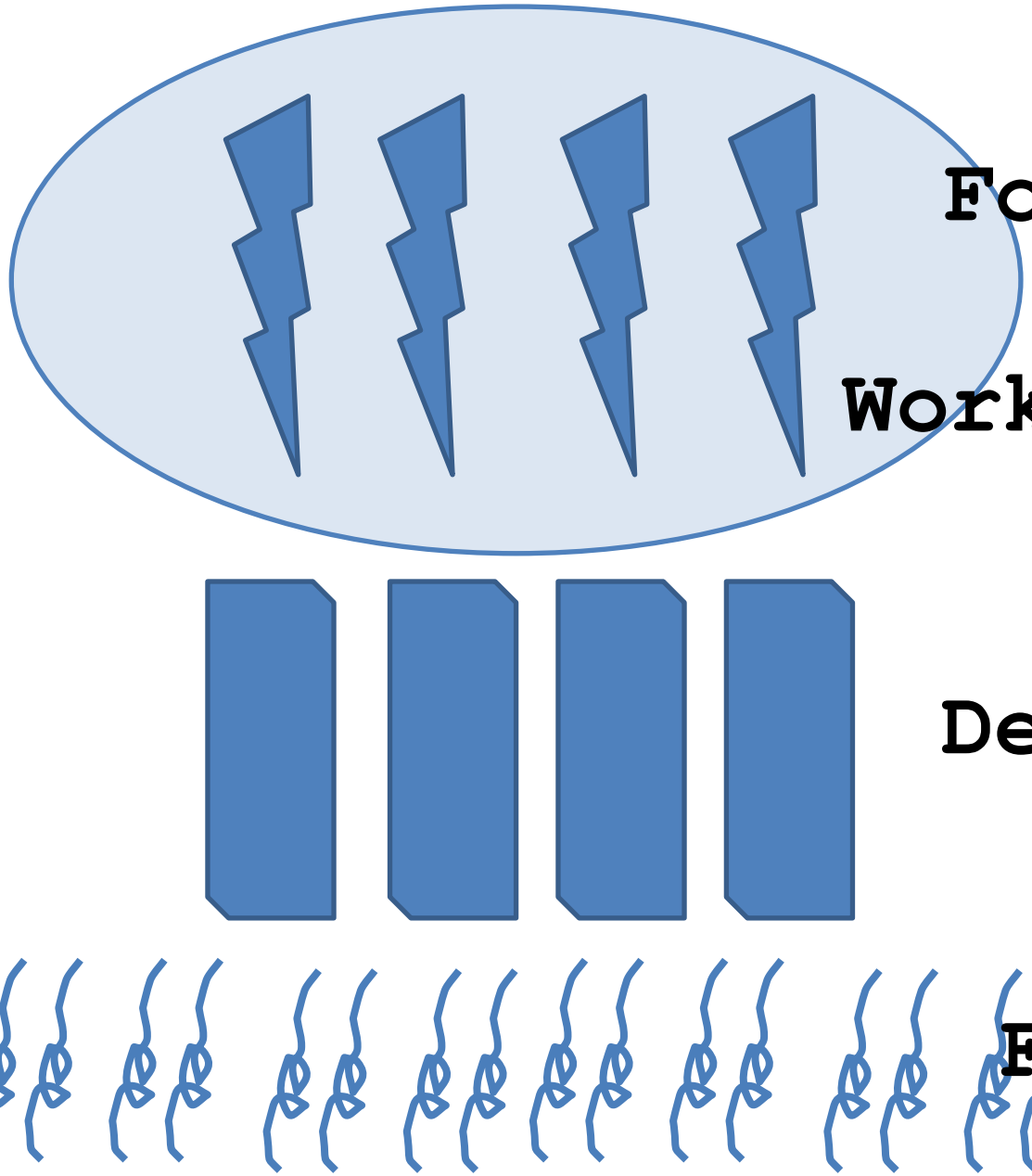
Vector is list of positions for repeated Strings

hash(key) -> index i

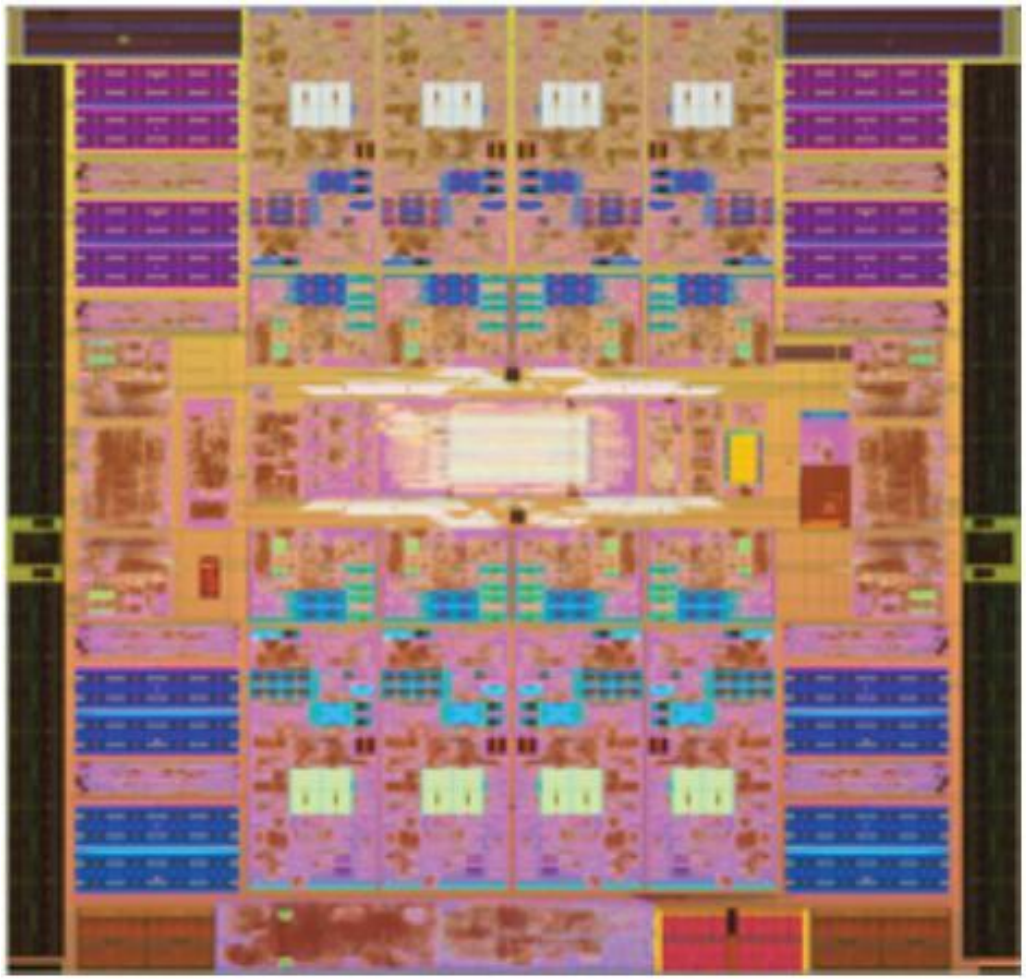





hash(key) -> index i
hash(key2) -> index j


ForkJoinPool
WorkerThreads
Dequeues
ForkJoinTasks

Sun UltraSPARC T2



CMP (8 cores)

SMT (8 threads)

UMA (4MB L2
32GB main)

Concordance algorithm

1) Read file into String array
2)
    a. Split into regions (with lookahead for N)
    b. Allocate one ForkJoinTask per region
    c. Each task finds phrases, adds index to global ConcurrentHashMap
3) Iterate over keys in HashMap, print out index vector

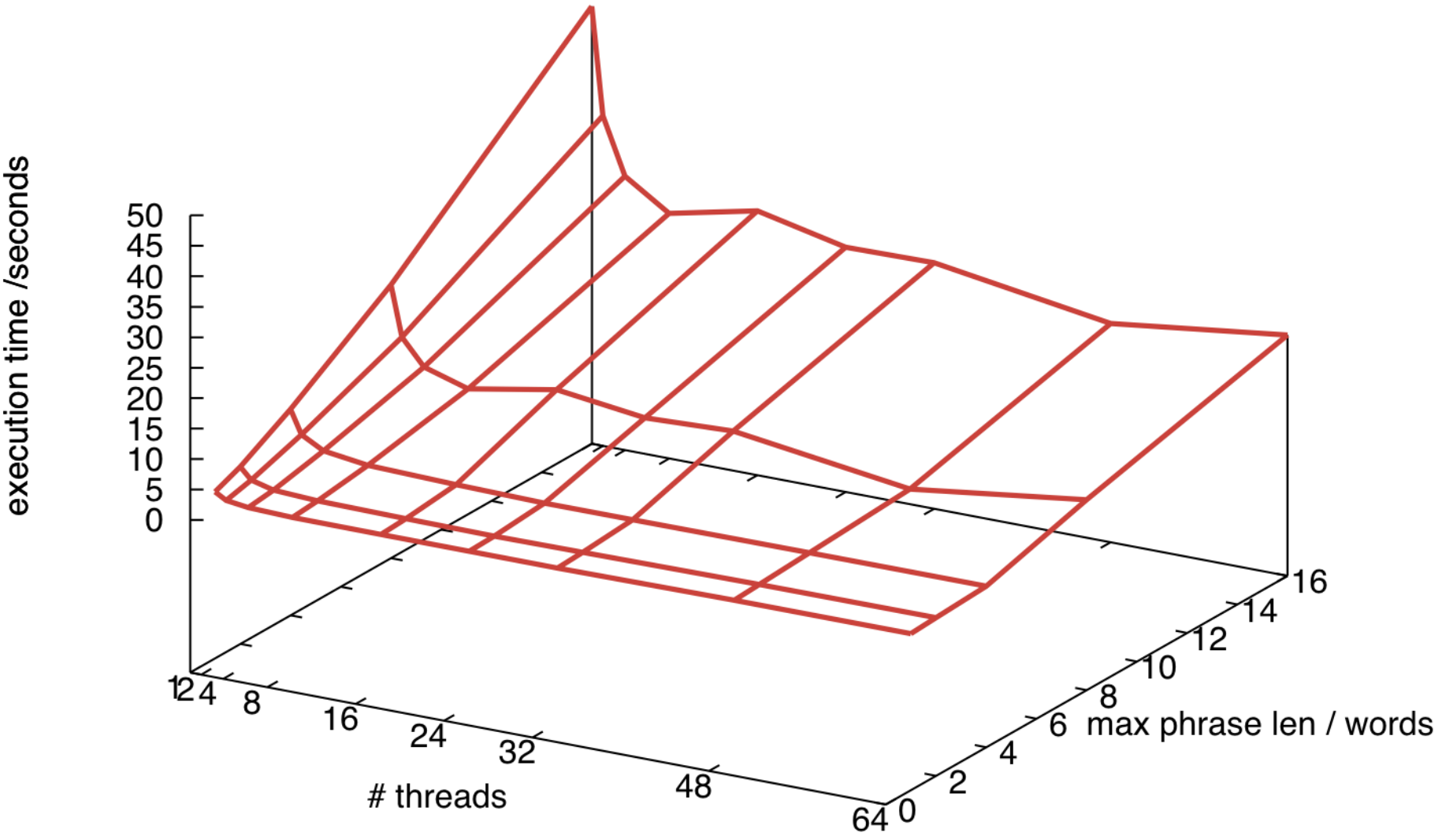
Timing breakdown

- 1) read 4MB file – 8.7s
- 2) create concordance for N=4 – 10.8s
- 3) print results (unsorted, 43MB) – 115s

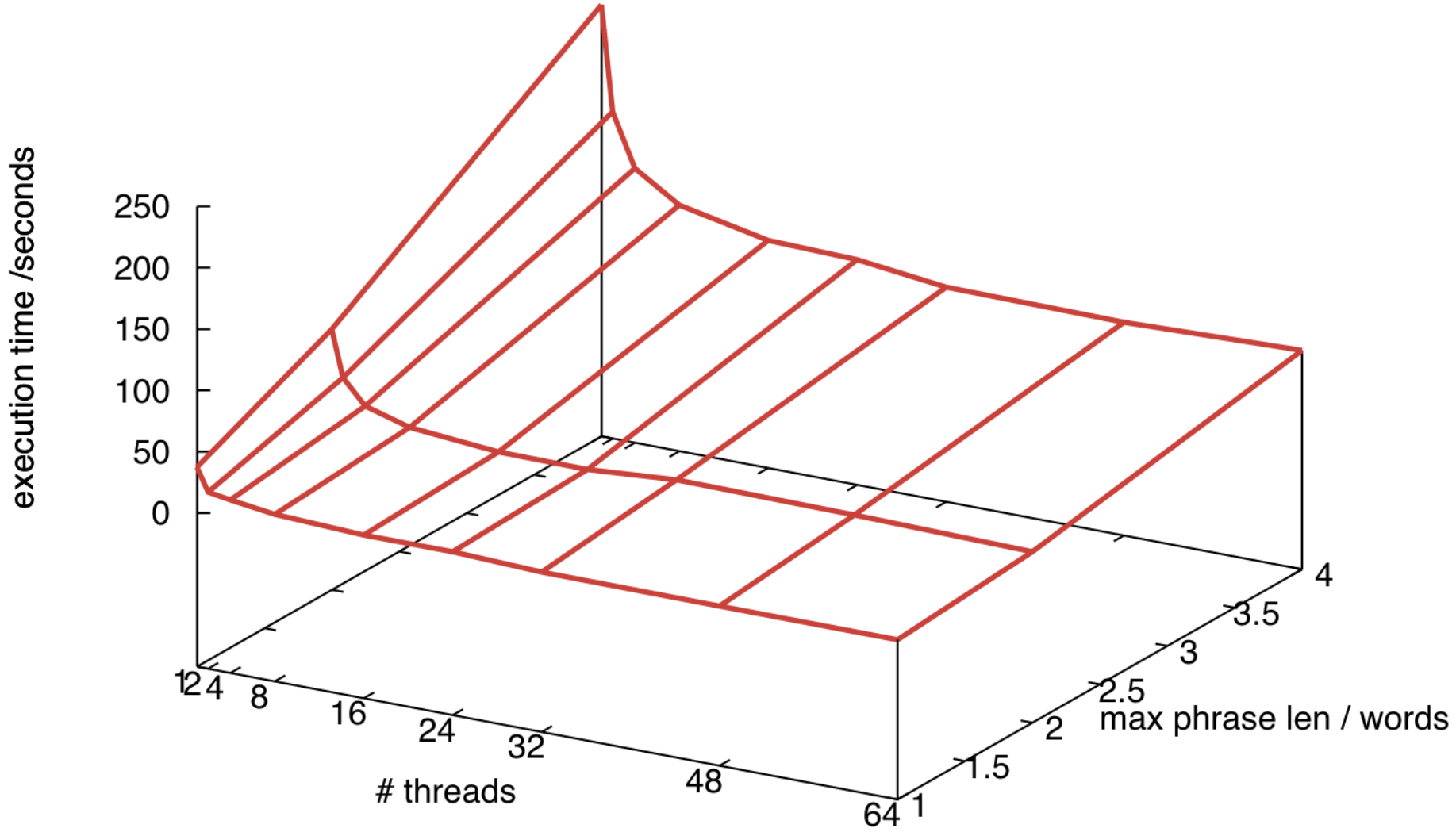
Timing issues

- Heap growth
- Garbage collection
- Number of tasks per worker thread
- Number of threads
- Contention in `ConcurrentHashMap`
- Size of `ConcurrentHashMap`

Exploring 2d parameter space (4MB)



Exploring 2d parameter space (100MB)

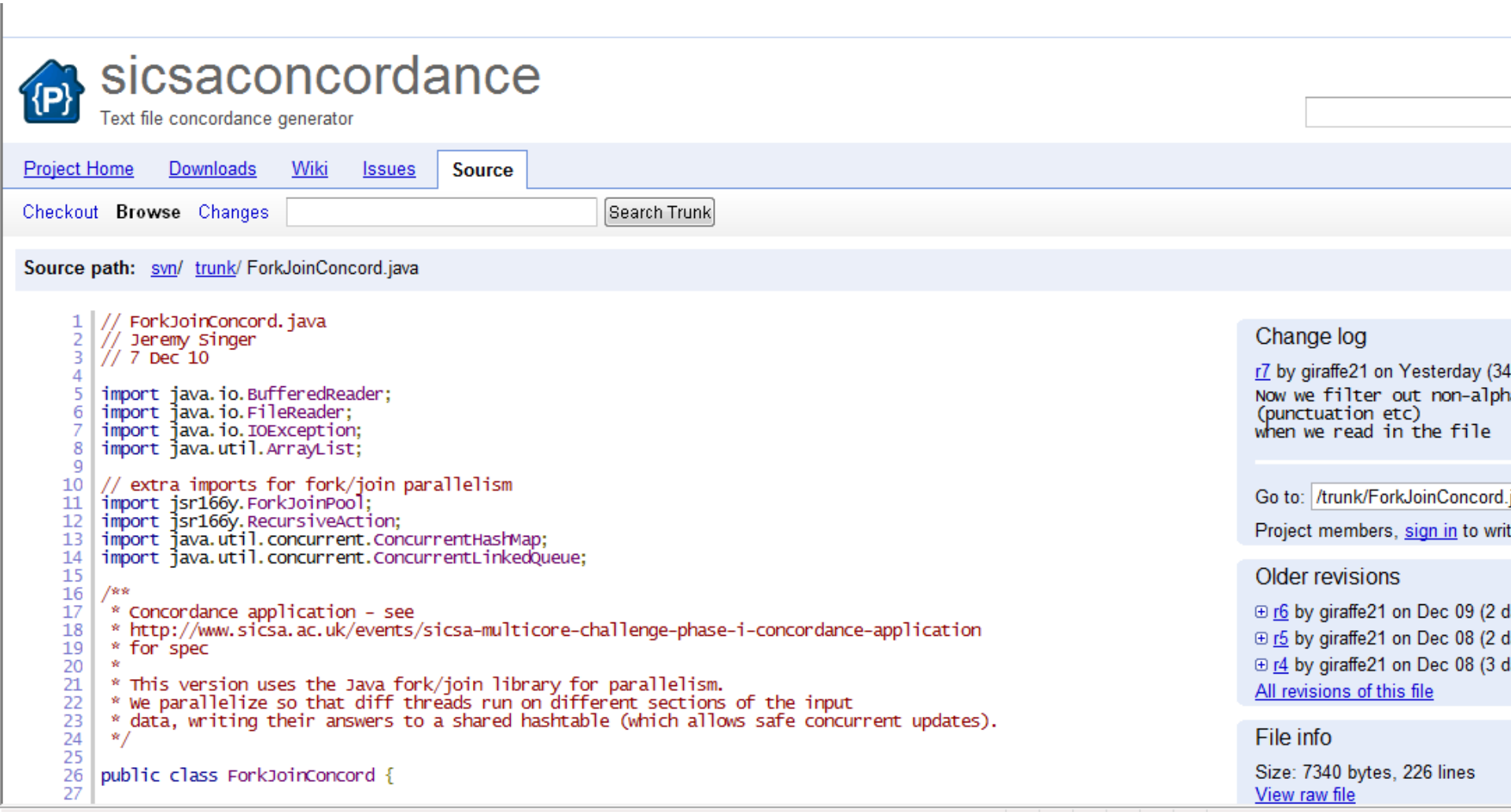


Further params to tune

- (see timing issues)

- Exhaustive exploration impossible
- Search-based auto-tuning techniques
  - Machine learning
  - (already do this for garbage collection config)

http://siscaconcordance.googlecode.com



sicsaconcordance

Text file concordance generator

Project Home | Downloads | Wiki | Issues | Source

Checkout | Browse | Changes | Search Trunk

Source path: svn/ trunk/ ForkJoinConcord.java

```
1  // ForkJoinConcord.java
2  // Jeremy Singer
3  // 7 Dec 10
4
5  import java.io.BufferedReader;
6  import java.io.FileReader;
7  import java.io.IOException;
8  import java.util.ArrayList;
9
10 // extra imports for fork/join parallelism
11 import jsr166y.ForkJoinPool;
12 import jsr166y.RecursiveAction;
13 import java.util.concurrent.ConcurrentHashMap;
14 import java.util.concurrent.ConcurrentLinkedQueue;
15
16 /**
17  * Concordance application - see
18  * http://www.sicsa.ac.uk/events/sicsa-multicore-challenge-phase-i-concordance-application
19  * for spec
20  *
21  * This version uses the Java fork/join library for parallelism.
22  * We parallelize so that diff threads run on different sections of the input
23  * data, writing their answers to a shared hashtable (which allows safe concurrent updates).
24  */
25
26 public class ForkJoinConcord {
27
```

Change log

r7 by giraffe21 on Yesterday (34

Now we filter out non-alph
(punctuation etc)
when we read in the file

Go to: /trunk/ForkJoinConcord.

Project members, sign in to writ

Older revisions

r6 by giraffe21 on Dec 09 (2 d
r5 by giraffe21 on Dec 08 (2 d
r4 by giraffe21 on Dec 08 (3 d
All revisions of this file

File info

Size: 7340 bytes, 226 lines
View raw file