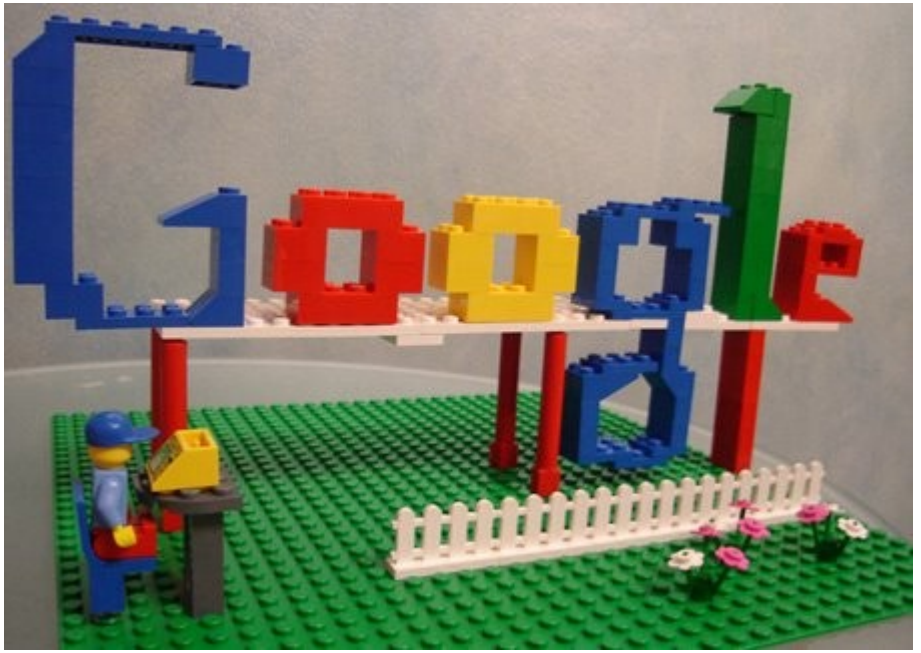
Who is scared of google?

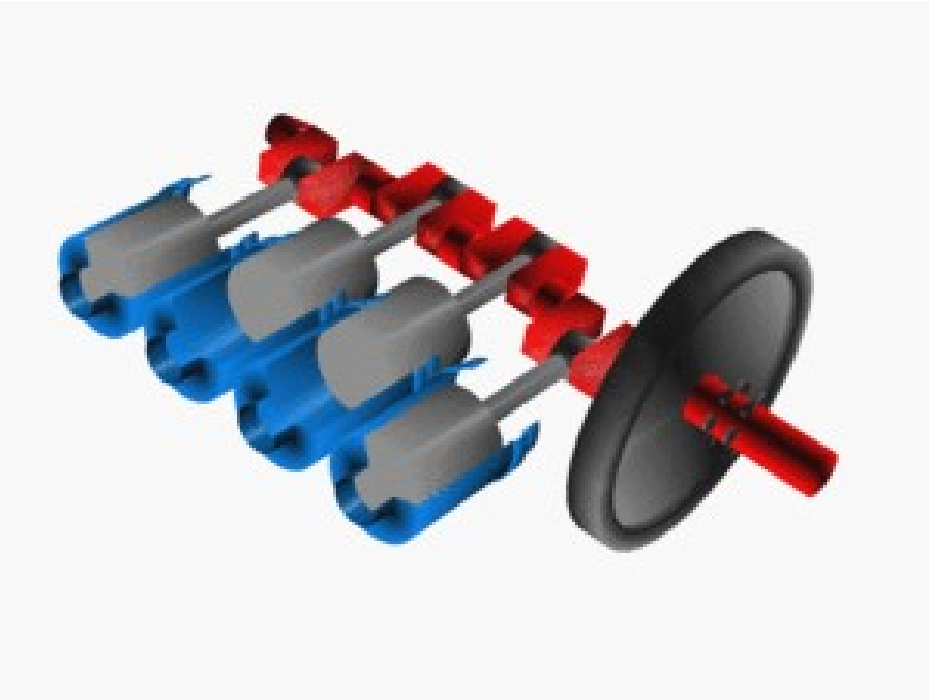


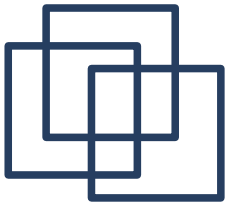
Steam Engine of this age

- Just as raw as steam engines were in the18th century
- Generically used everywhere
- Popular theme of its times

So what happened

- Only Crank-Shaft Design survived of what was the steam-engine by end of 19th century.
- New and powerful models of the same evolved to be engines of today



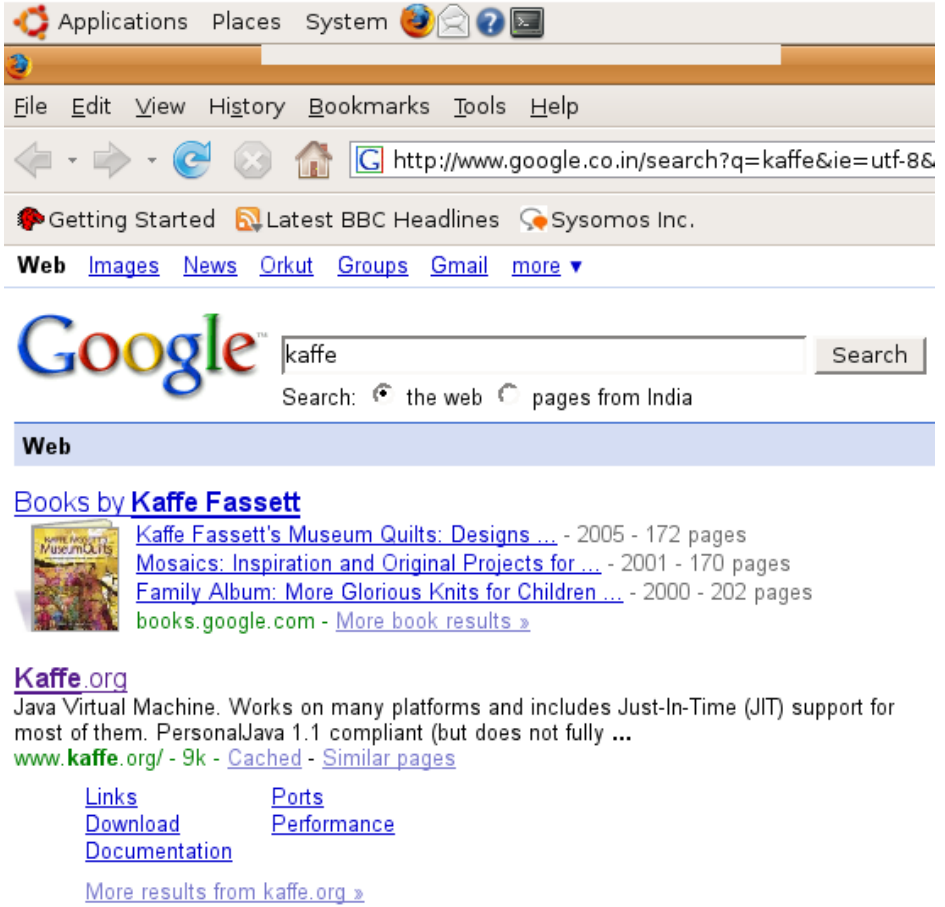


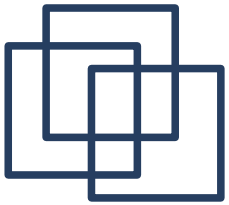
What am I saying?

- Google or any search engine today will not survive in its' current state in next 2 years.
- A mesh of specialized search engines is going to emerge which will have only the 'Crank-Shaft' design in common with what is today 'Google'.

It is already happening

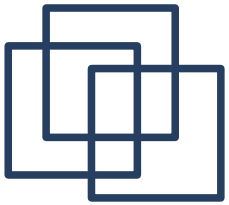
- Books are recommended
- Site specific pages are shown
- There will surely be more such features





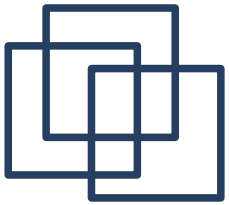
What has it got to do with me?

- You can come up with your own.
- Most of the technology behind this is available in FOSS. The 'Crank-Shaft' is luckily not owned by anyone!
- Lucene, Xapian, Lustre, Hadoop, Heritrix, MySQL, Nagios, Linux... cover a whole gamut of technology from server platform to indexing and search logic.



Kutti – a Sample Search

- It makes an index of Hindi Wikipedia
- Searches through this index



Indexing data

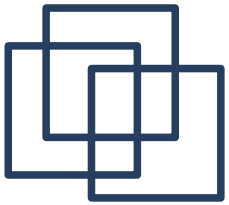
```
WORD_RE = re.compile(u"[\u0901-\u097Fa-zA-Z0-9]+", re.U)
class indexer:
   def __init__(self, index, feed):
      try:
         self.db = xapian.WritableDatabase(index, xapian.DB_CREATE_OR_OPEN)
      except Exception, e:
         print 'db not open'
      self.feed = MediaXML(feed)
      self.indexer = xapian.TermGenerator()
      self.stemmer = xapian.Stem('english')
      self.indexer.set_stemmer(self.stemmer)
      self.counter = 0
   def run(self):
      for title, text in self.feed.next():
         doc = xapian.Document()
         if text != None:
            for index, term in enumerate(WORD_RE.finditer(text)):
               if len(term.group()) > 60: continue
               doc.add_posting(term.group(),
                           index)


         for index, term in enumerate(WORD_RE.finditer(title)):
            doc.add_posting(term.group(), index)
        # print "TITLE:%s" % title
         doc.set_data(title)
         self.counter = self.counter + 1
         self.db.add_document(doc)
         print '----------'
      return 0
```

Opening Search Index

Word tokens

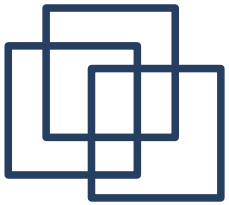
Adding tokens into a document

Writing document into an index



The Indexing Process

- A document is an indexable entity eg. webpage or item in xml or a row in database
- A document is broken into a stream of tokens
- Tokens are indexed into a search database



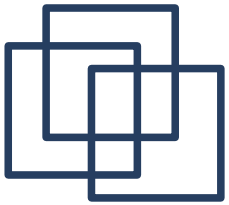
Search logic

```
database = xapian.Database(sys.argv[1])

# Start an enquire session.
enquire = xapian.Enquire(database)

query_string = sys.argv[2]
for arg in sys.argv[3:]:
    query_string += ' '
    query_string += arg

# Parse the query string to produce a Xapian::Query object.
qp = xapian.QueryParser()
stemmer = xapian.Stem("english")
qp.set_stemmer(stemmer)
qp.set_database(database)
qp.set_stemming_strategy(xapian.QueryParser.STEM_NONE)
query = qp.parse_query(query_string)
print "Parsed query is: %s" % query.get_description()

# Find the top 10 results for the query.
enquire.set_query(query)
matches = enquire.get_mset(0, 10)

# Display the results.
print "%i results found." % matches.get_matches_estimated()
print "Results 1-%i:" % matches.size()

for m in matches:
    print "%i: %i%% docid=%i [%s]" % (m[xapian.MSET_RANK] + 1, m[xapian.MSET_PERCENT], m[xapian.MSET_DID],
 m[xapian.MSET_DOCUMENT].get_data())
```

Opening search database

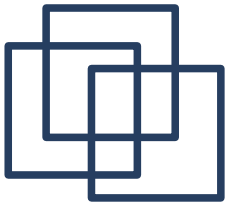
Query parsing

Searching



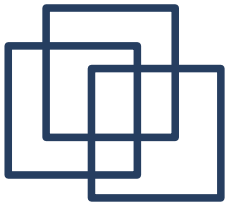
The Search Process

- A query is parsed into tokens similar to the process used for indexing
- The parsed tokens are searched in a search database
- Documents found with those tokens are retrieved



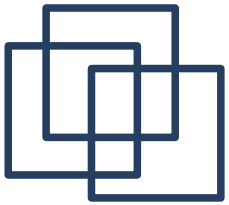
Few pet projects

- Indic search of wikipedia
- Mis-spelled queries
- Mesh of search servers with DHT
- Mobile search
- Automatic tagging of text
- Code Searching



If you still have questions…

- Pay me :)
- Pay Doug Cutting
- Google it



Contact

supreet.sethi+OSIW-queries@gmail.com