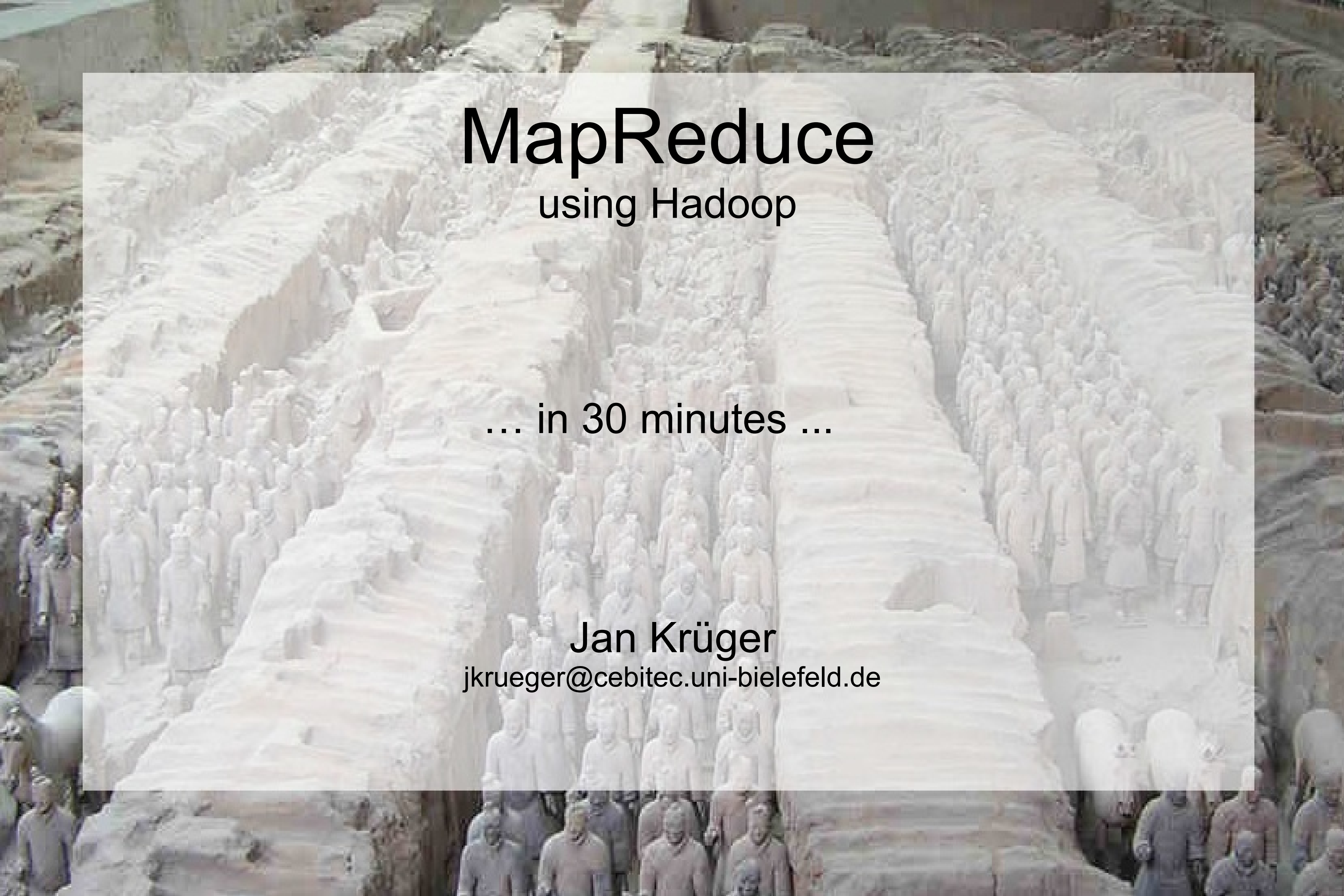
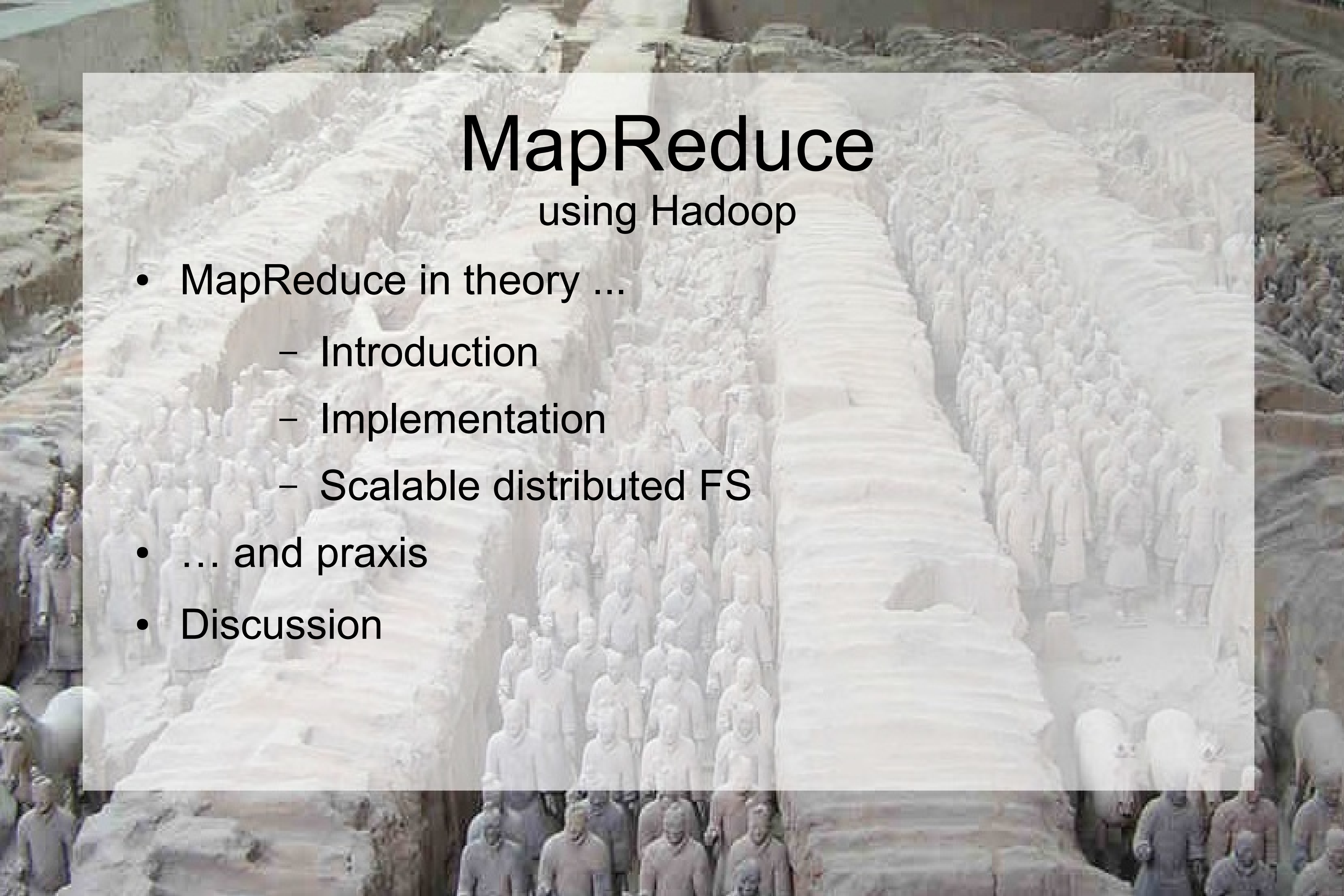


MapReduce

using Hadoop

… in 30 minutes ...

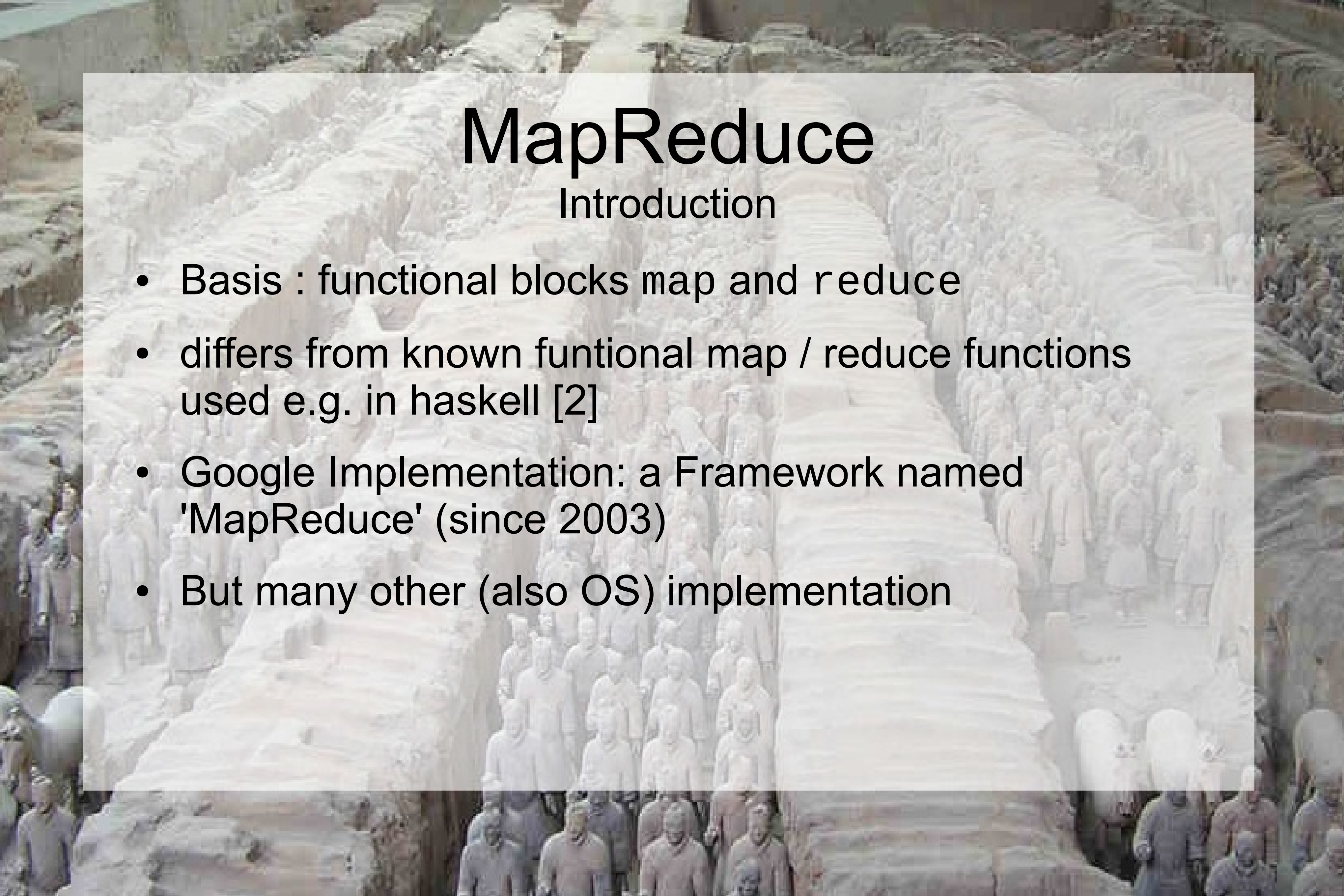
Jan Krüger
jkrueger@cebitec.uni-bielefeld.de



MapReduce

using Hadoop

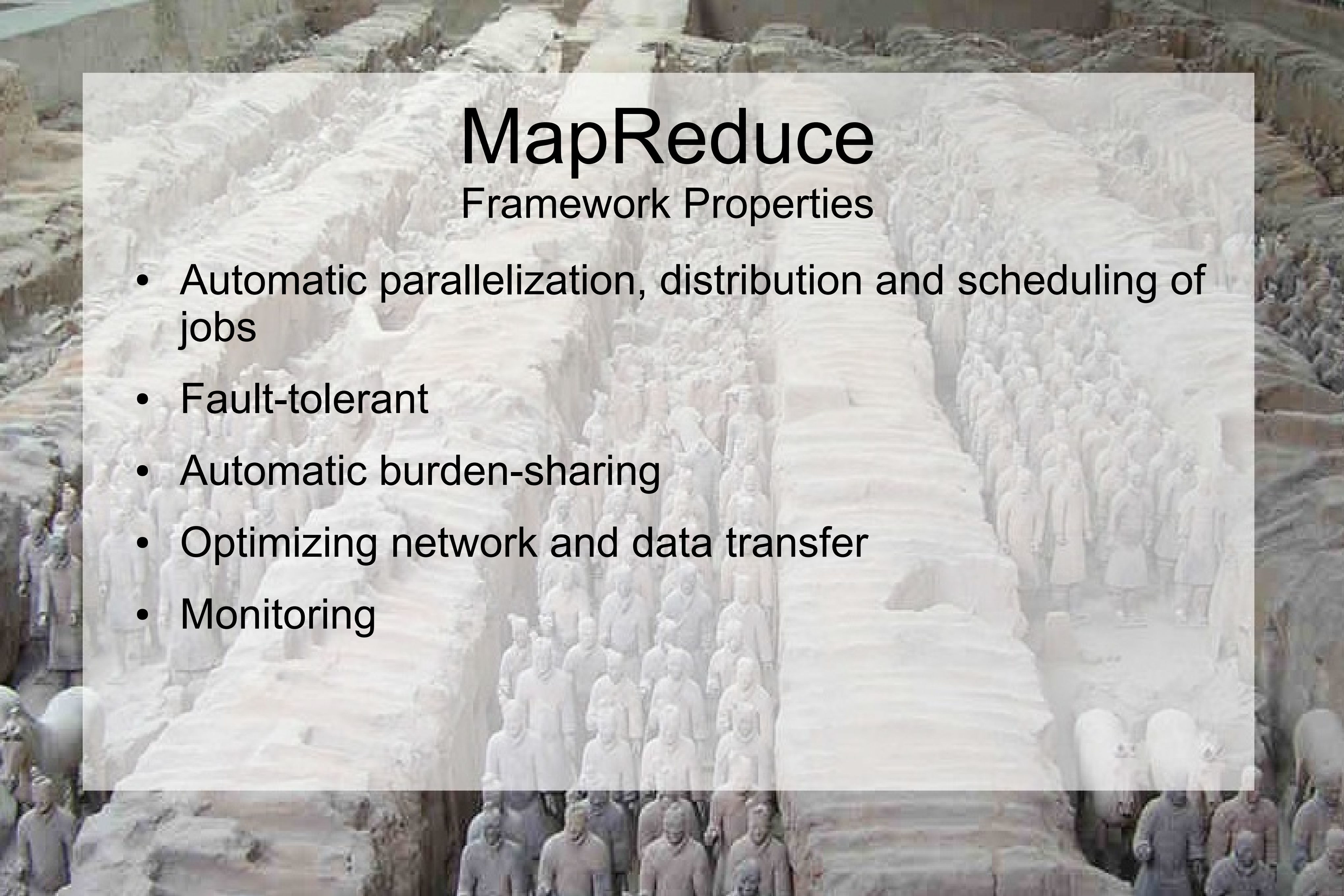
- MapReduce in theory ...
  - Introduction
  - Implementation
  - Scalable distributed FS
- … and praxis
- Discussion



MapReduce

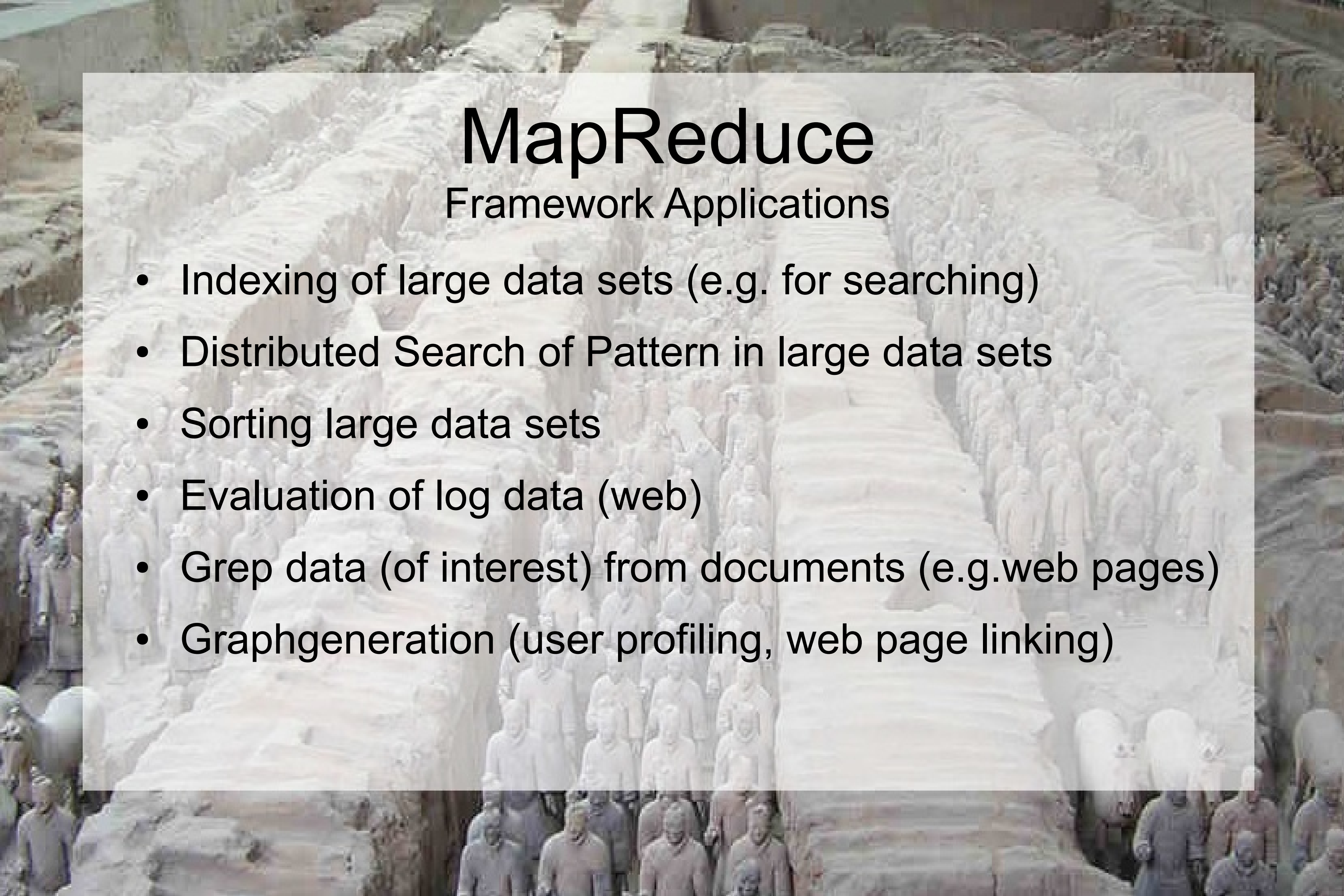
Introduction

- Basis : functional blocks `map` and `reduce`
- differs from known funtional map / reduce functions used e.g. in haskell [2]
- Google Implementation: a Framework named 'MapReduce' (since 2003)
- But many other (also OS) implementation



MapReduce
Framework Properties

- Automatic parallelization, distribution and scheduling of jobs
- Fault-tolerant
- Automatic burden-sharing
- Optimizing network and data transfer
- Monitoring



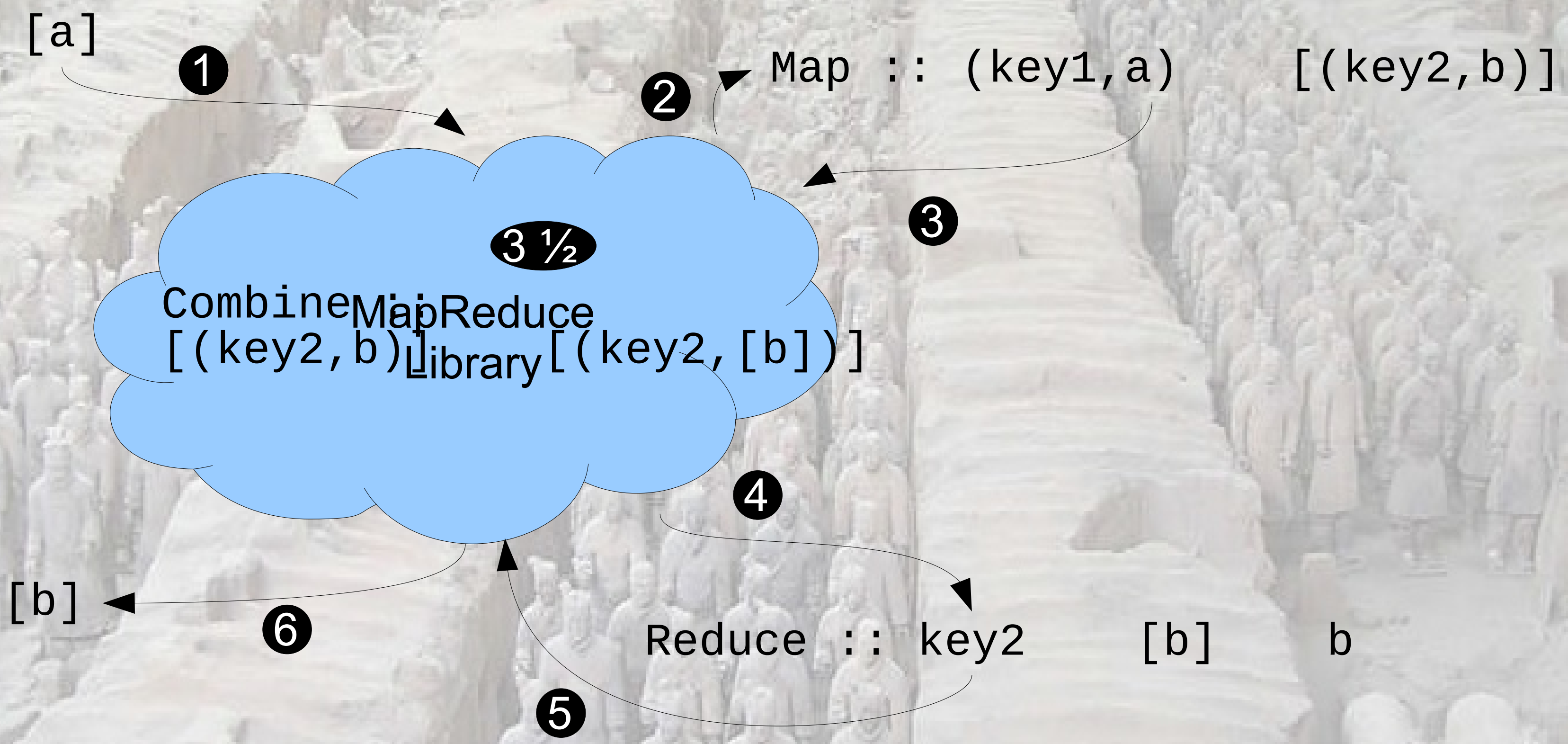
MapReduce

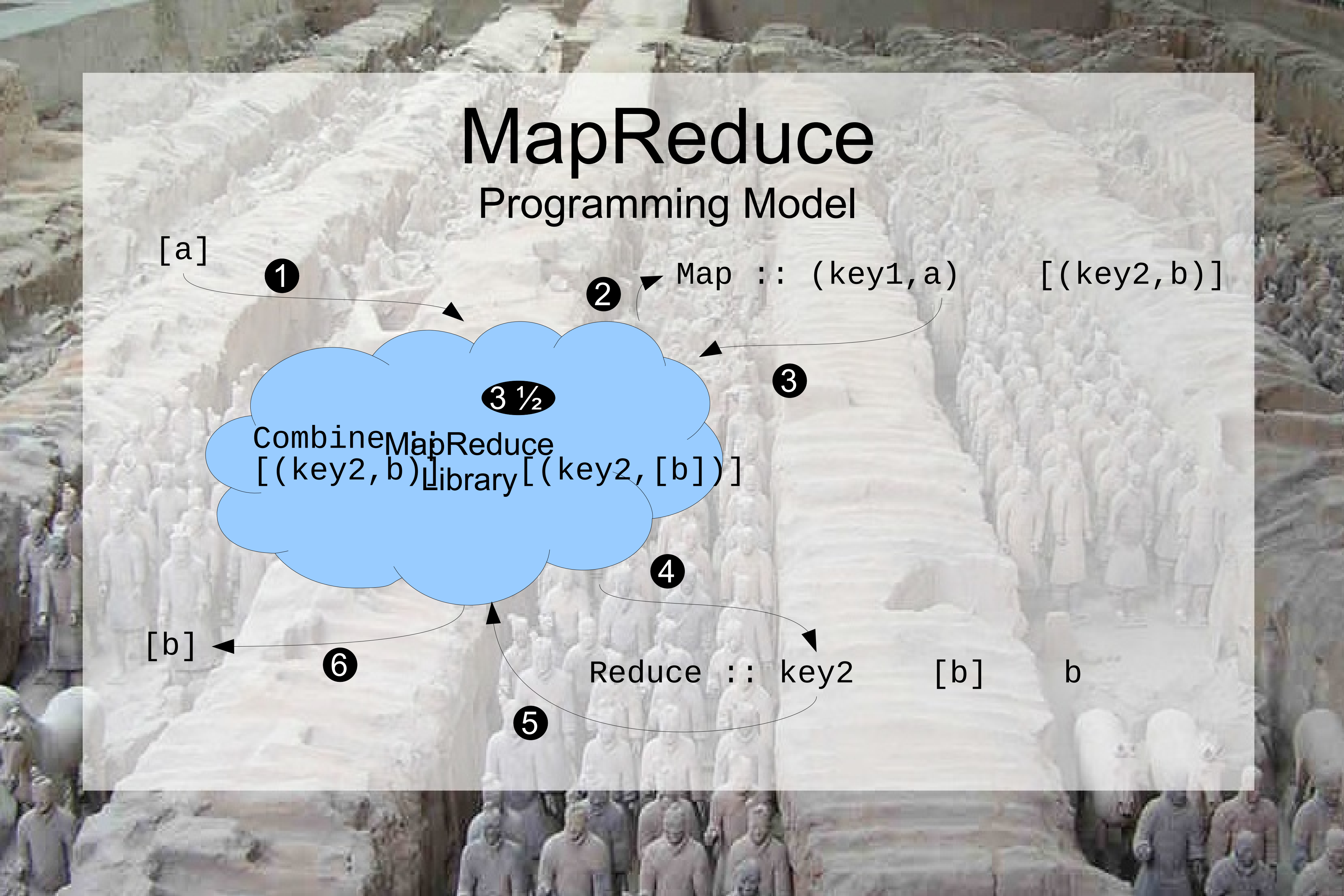
Framework Applications

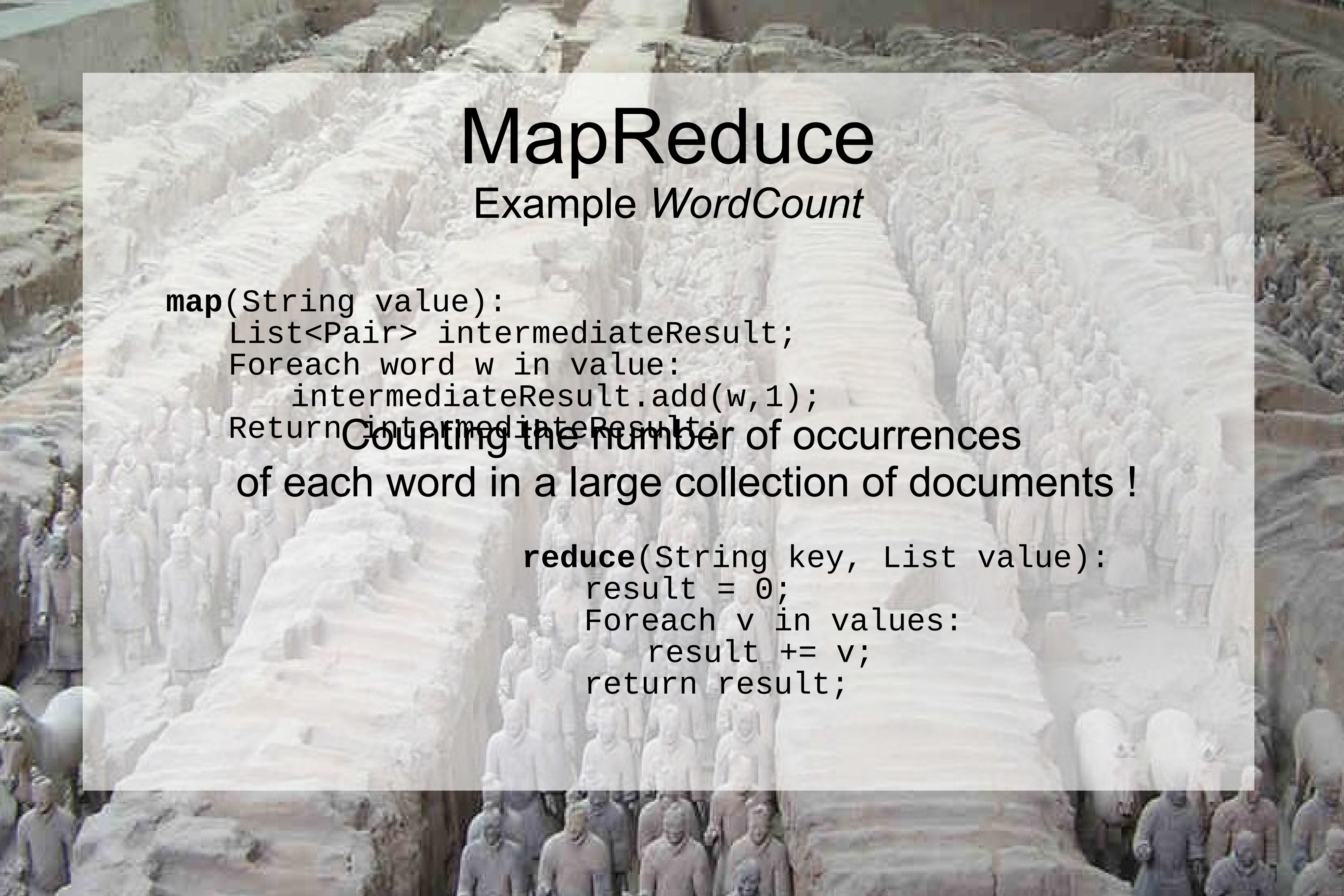
- Indexing of large data sets (e.g. for searching)
- Distributed Search of Pattern in large data sets
- Sorting large data sets
- Evaluation of log data (web)
- Grep data (of interest) from documents (e.g.web pages)
- Graphgeneration (user profiling, web page linking)

MapReduce

Programming Model







MapReduce

Example *WordCount*

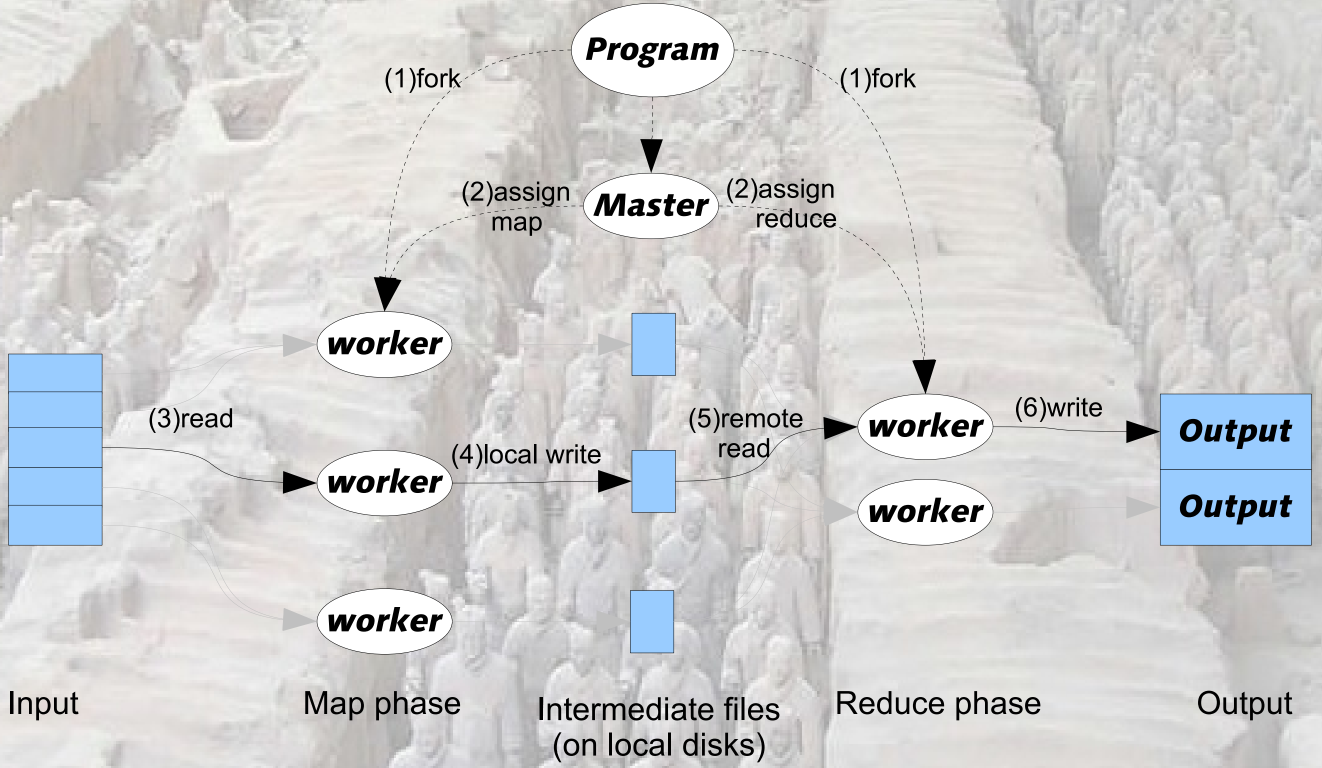
```
map(String value):
    List<Pair> intermediateResult;
    Foreach word w in value:
        intermediateResult.add(w,1);
    Return intermediateResult;
```

Counting the number of occurrences
of each word in a large collection of documents !

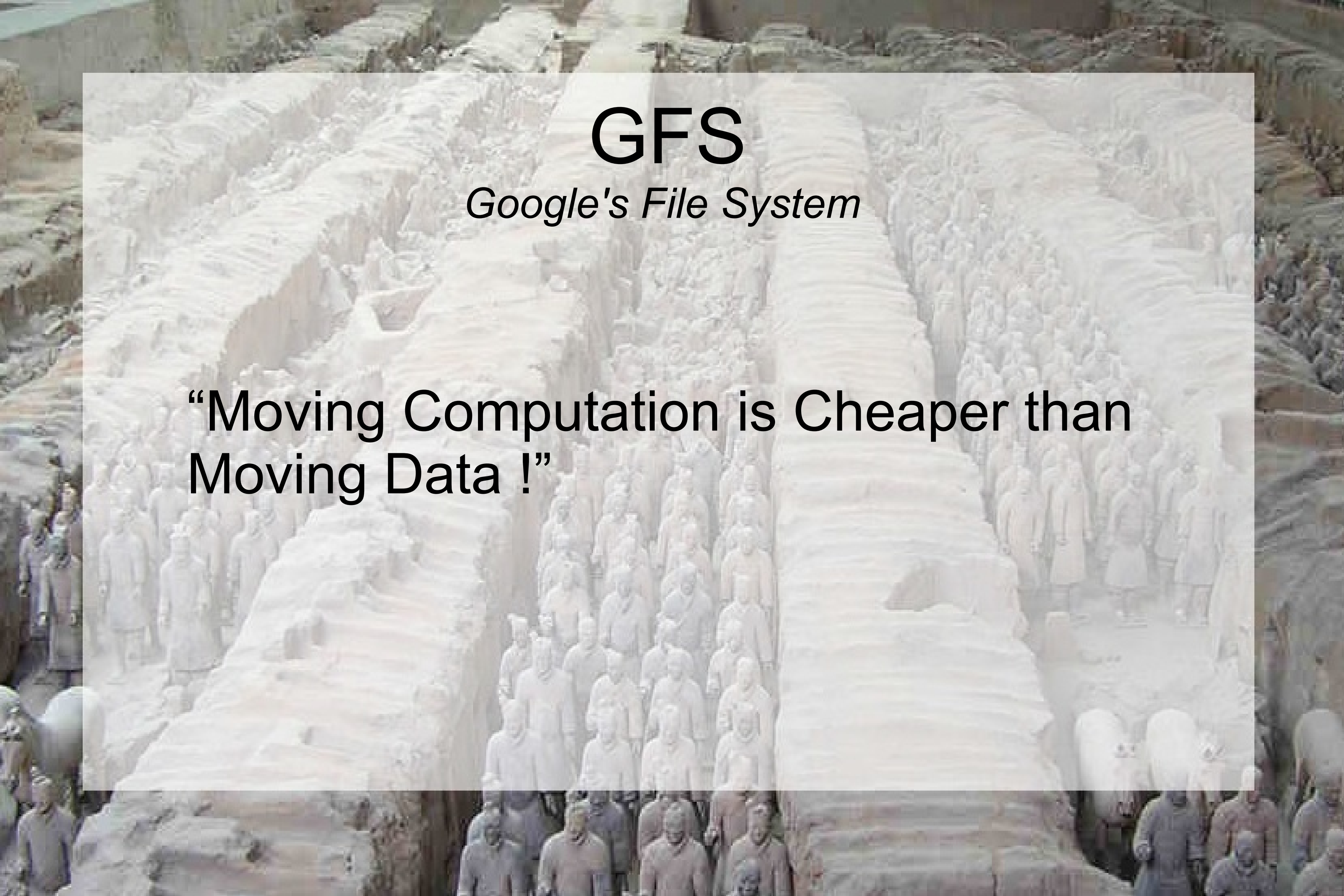
```
reduce(String key, List value):
    result = 0;
    Foreach v in values:
        result += v;
    return result;
```

MapReduce

Google's Implementation



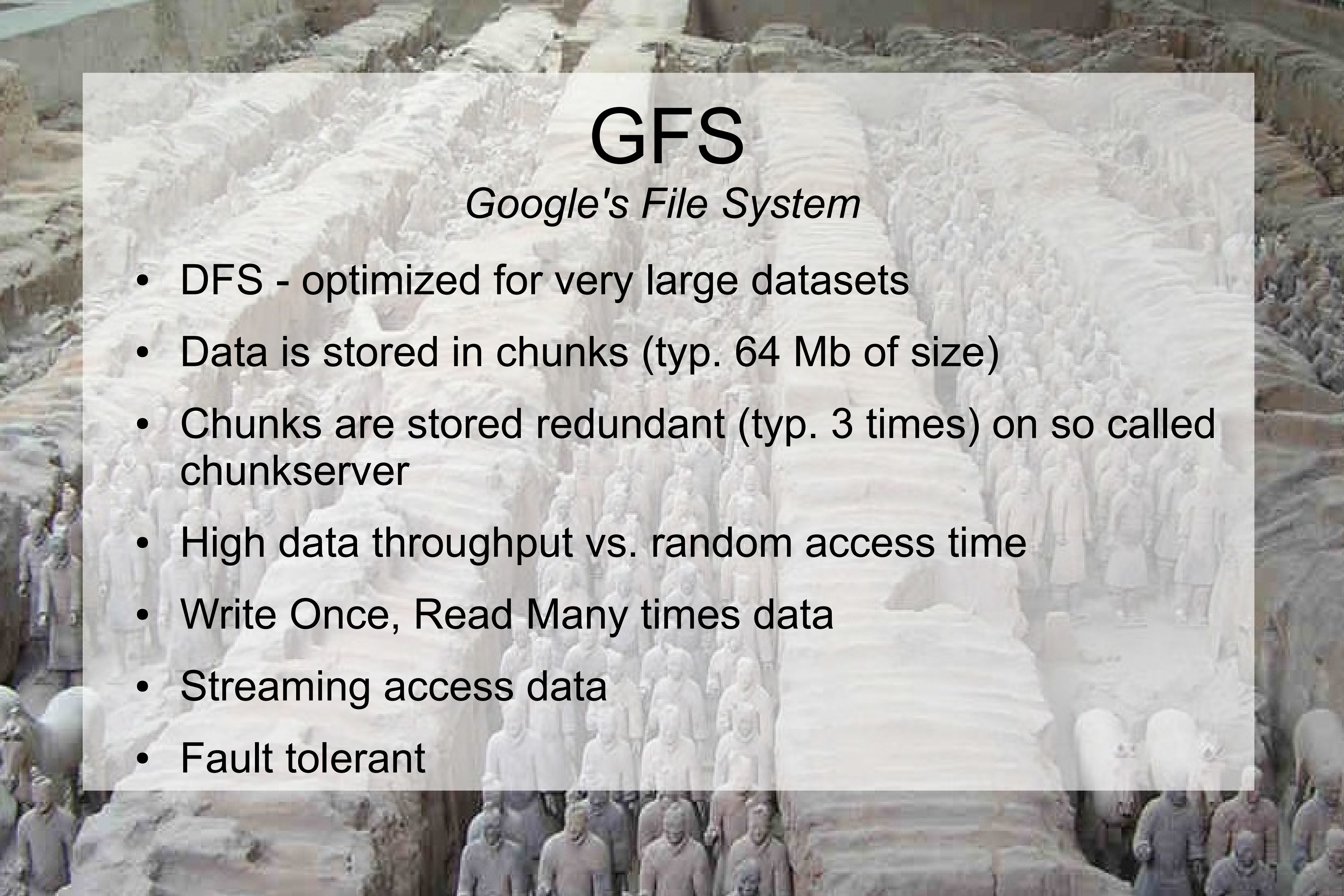
Execution Overview taken from [1]



GFS

Google's File System

“Moving Computation is Cheaper than Moving Data !”



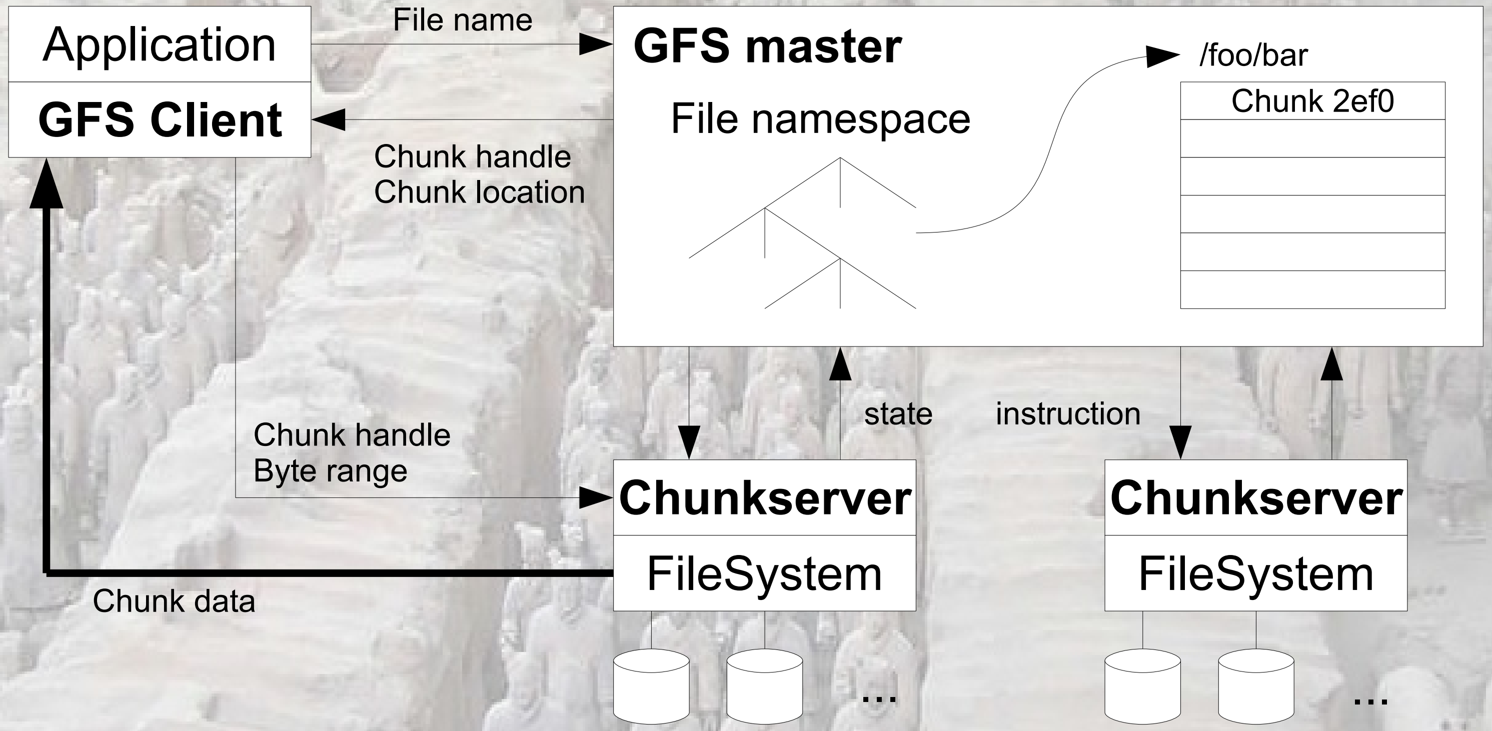
GFS

Google's File System

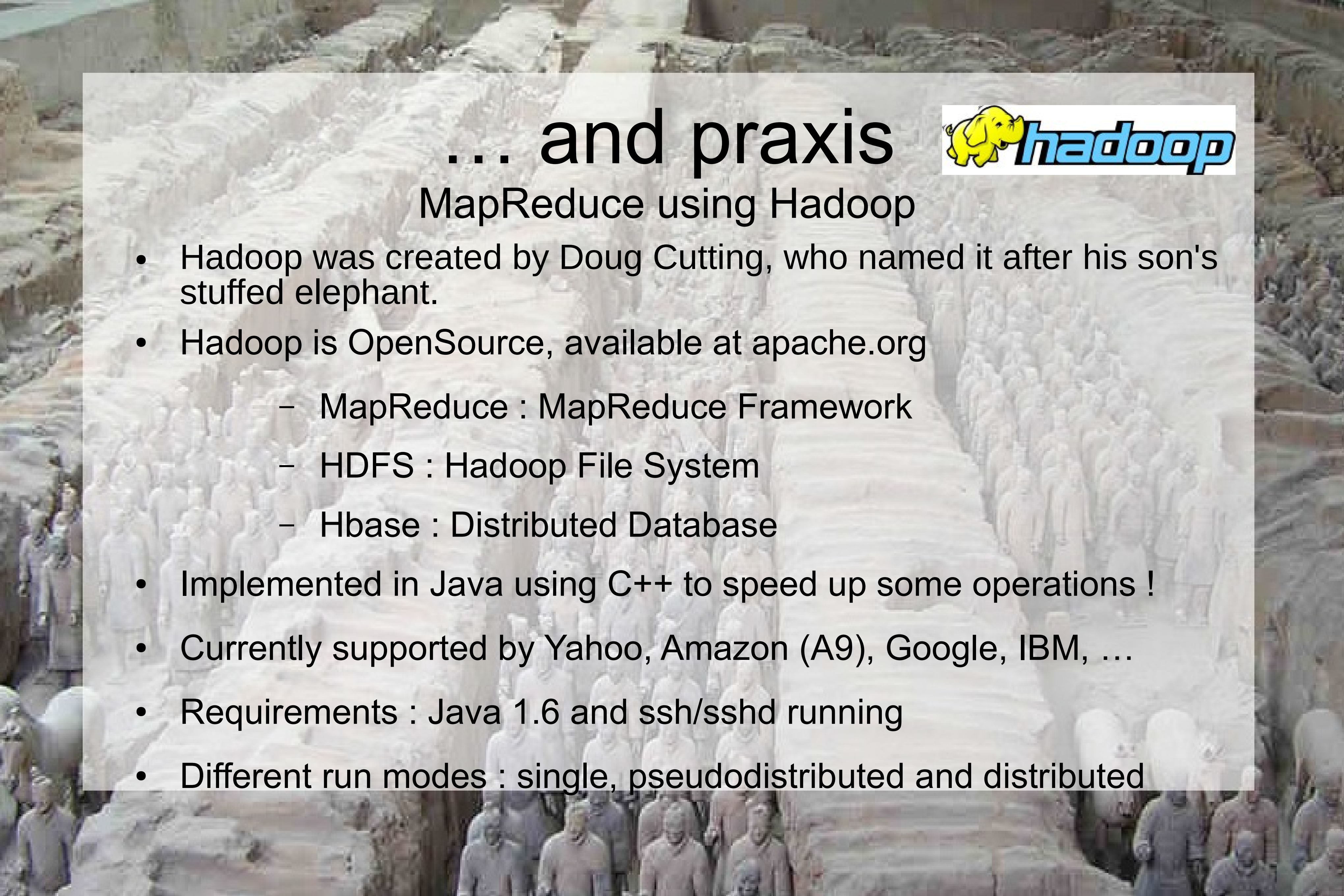
- DFS - optimized for very large datasets
- Data is stored in chunks (typ. 64 Mb of size)
- Chunks are stored redundant (typ. 3 times) on so called chunkserver
- High data throughput vs. random access time
- Write Once, Read Many times data
- Streaming access data
- Fault tolerant

GFS

Architecture



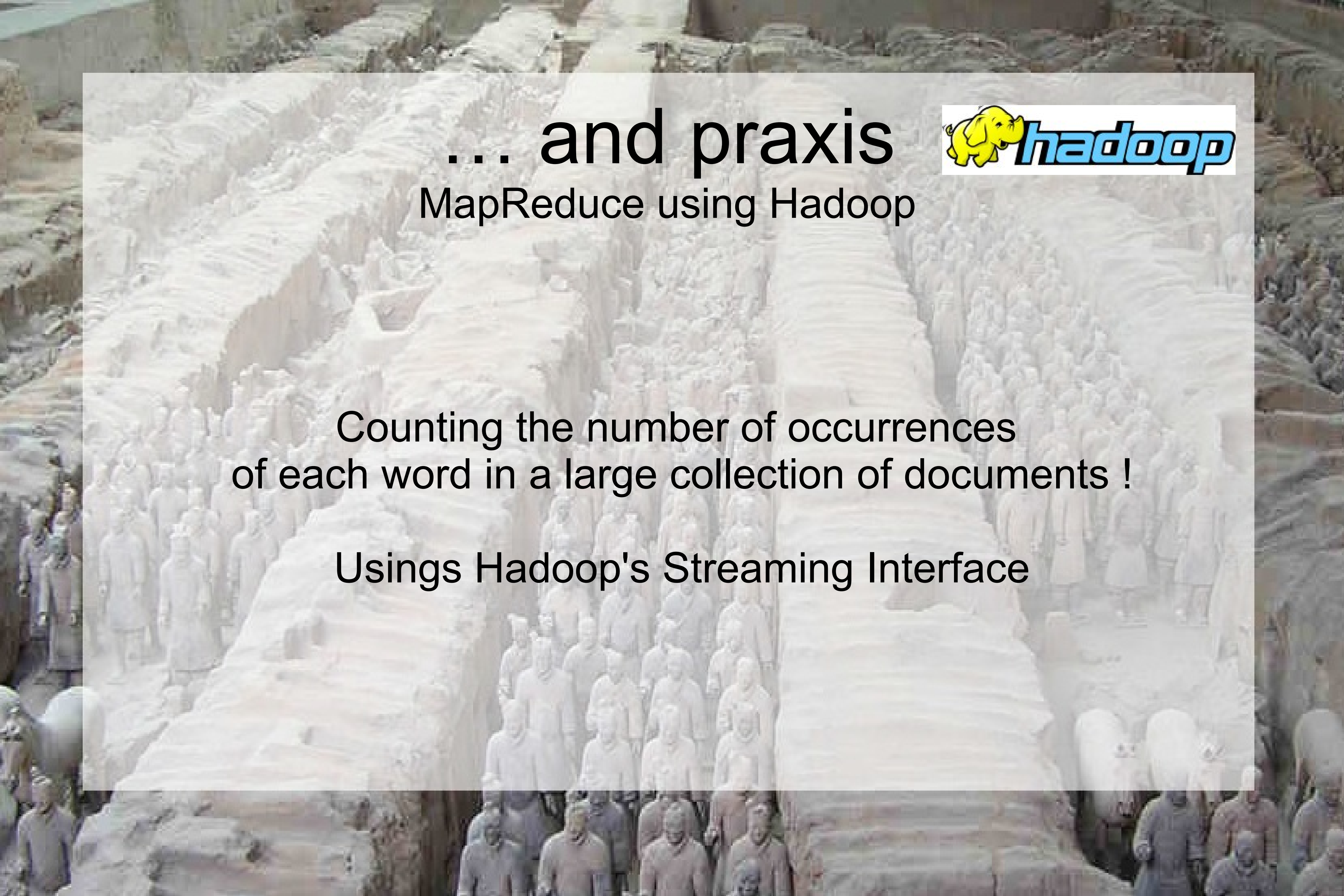
Architecture taken from [3]



… and praxis

MapReduce using Hadoop

- Hadoop was created by Doug Cutting, who named it after his son's stuffed elephant.
- Hadoop is OpenSource, available at apache.org
  - MapReduce : MapReduce Framework
  - HDFS : Hadoop File System
  - Hbase : Distributed Database
- Implemented in Java using C++ to speed up some operations !
- Currently supported by Yahoo, Amazon (A9), Google, IBM, …
- Requirements : Java 1.6 and ssh/sshd running
- Different run modes : single, pseudodistributed and distributed



… and praxis

MapReduce using Hadoop

Counting the number of occurrences
of each word in a large collection of documents !

Usings Hadoop's Streaming Interface

… and praxis

Python Mapper

```
#!/usr/bin/env python.
import ioMapReduce.
.
def mapper(key,value):.
    words = value.split().
    for word in words:.
        print "%s %s" % (word, 1).
.
ioMapReduce.readInput(mapper).
```

```
import sys.
.
def readInput(mapper):.
  for (num,line) in enumerate(sys.stdin):.
    line=line.strip().
    mapper(num,line).
```

... and praxis

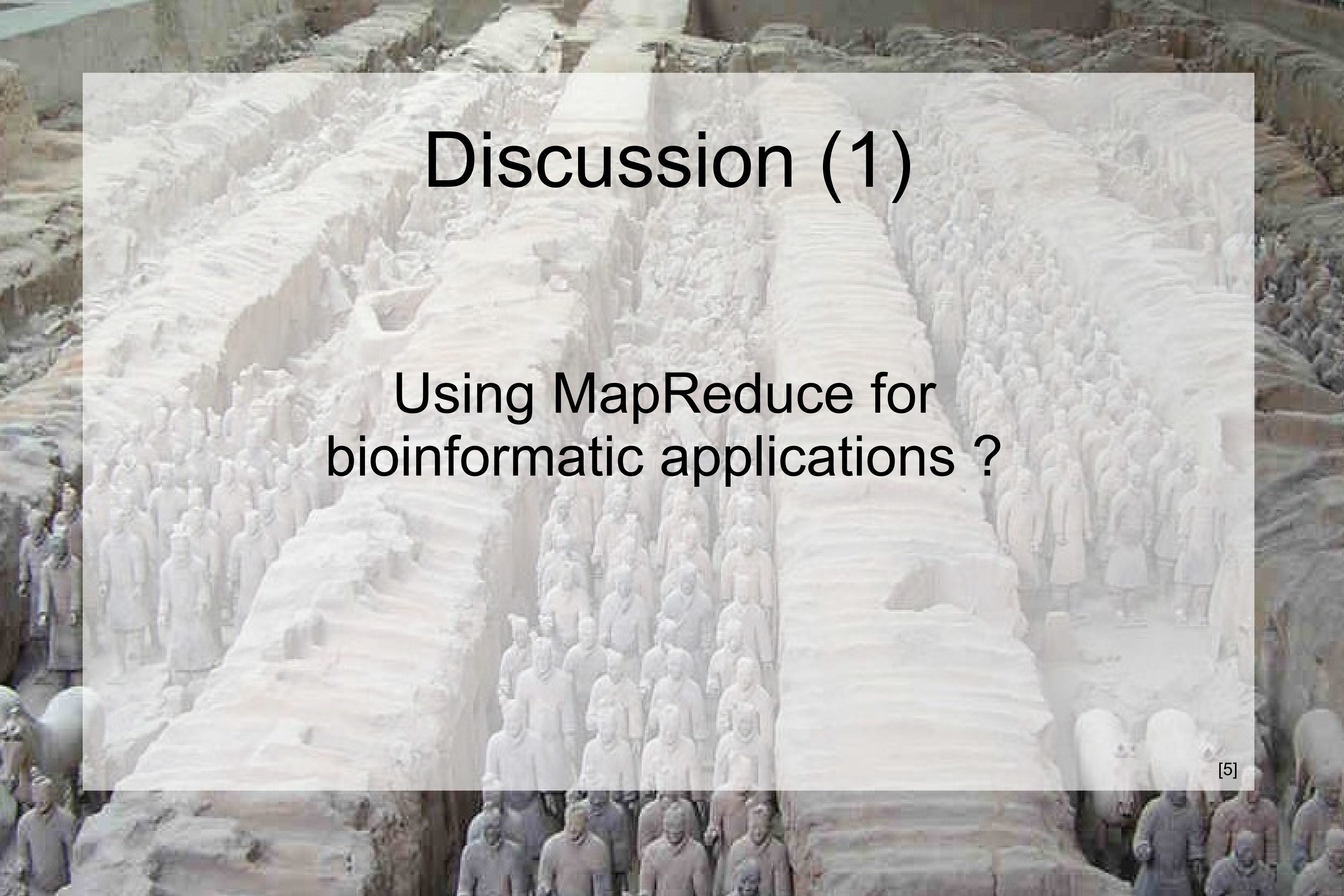
Python Reducer

```python
#!/usr/bin/env python
import sys
import ioMapReduce

def reducer(key,values):
  return len(values)

outputWriter(reducer)
```

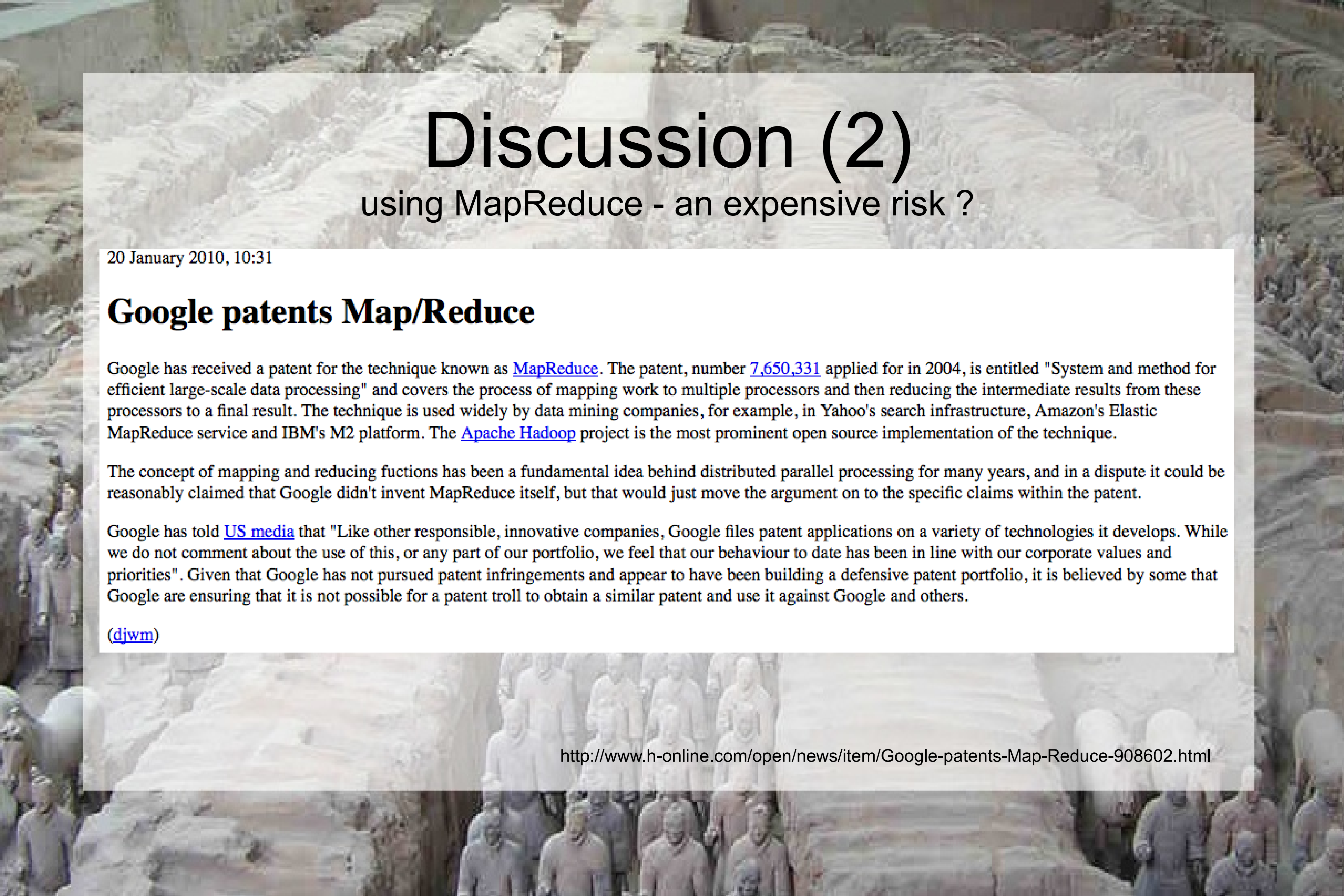
```python
def outputWriter(reducer):
  key2Values={}
  for line in sys.stdin:
    line= line.strip()
    key,value= line.split()
    try:
      key2Values[key].append(value)
    except:
      key2Values[key]=[value]
  results=[]
  for key in key2Values.keys():
    results.append((key,
                reducer(key,key2Values[key])))
  for (key,value)in results:
    print "%s %s"% (key,value)
```



Discussion (1)

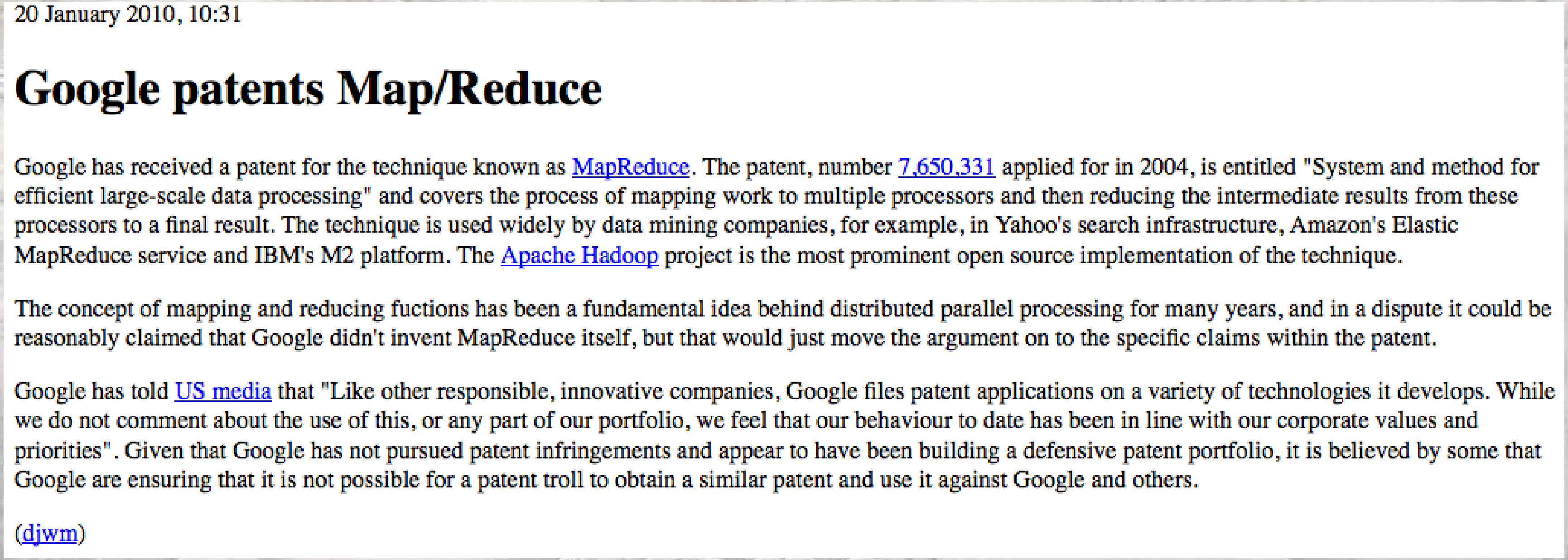
Using MapReduce for bioinformatic applications ?

[5]



Discussion (2)

using MapReduce - an expensive risk ?



20 January 2010, 10:31

Google patents Map/Reduce

Google has received a patent for the technique known as MapReduce. The patent, number 7,650,331 applied for in 2004, is entitled "System and method for efficient large-scale data processing" and covers the process of mapping work to multiple processors and then reducing the intermediate results from these processors to a final result. The technique is used widely by data mining companies, for example, in Yahoo's search infrastructure, Amazon's Elastic MapReduce service and IBM's M2 platform. The Apache Hadoop project is the most prominent open source implementation of the technique.

The concept of mapping and reducing fuctions has been a fundamental idea behind distributed parallel processing for many years, and in a dispute it could be reasonably claimed that Google didn't invent MapReduce itself, but that would just move the argument on to the specific claims within the patent.

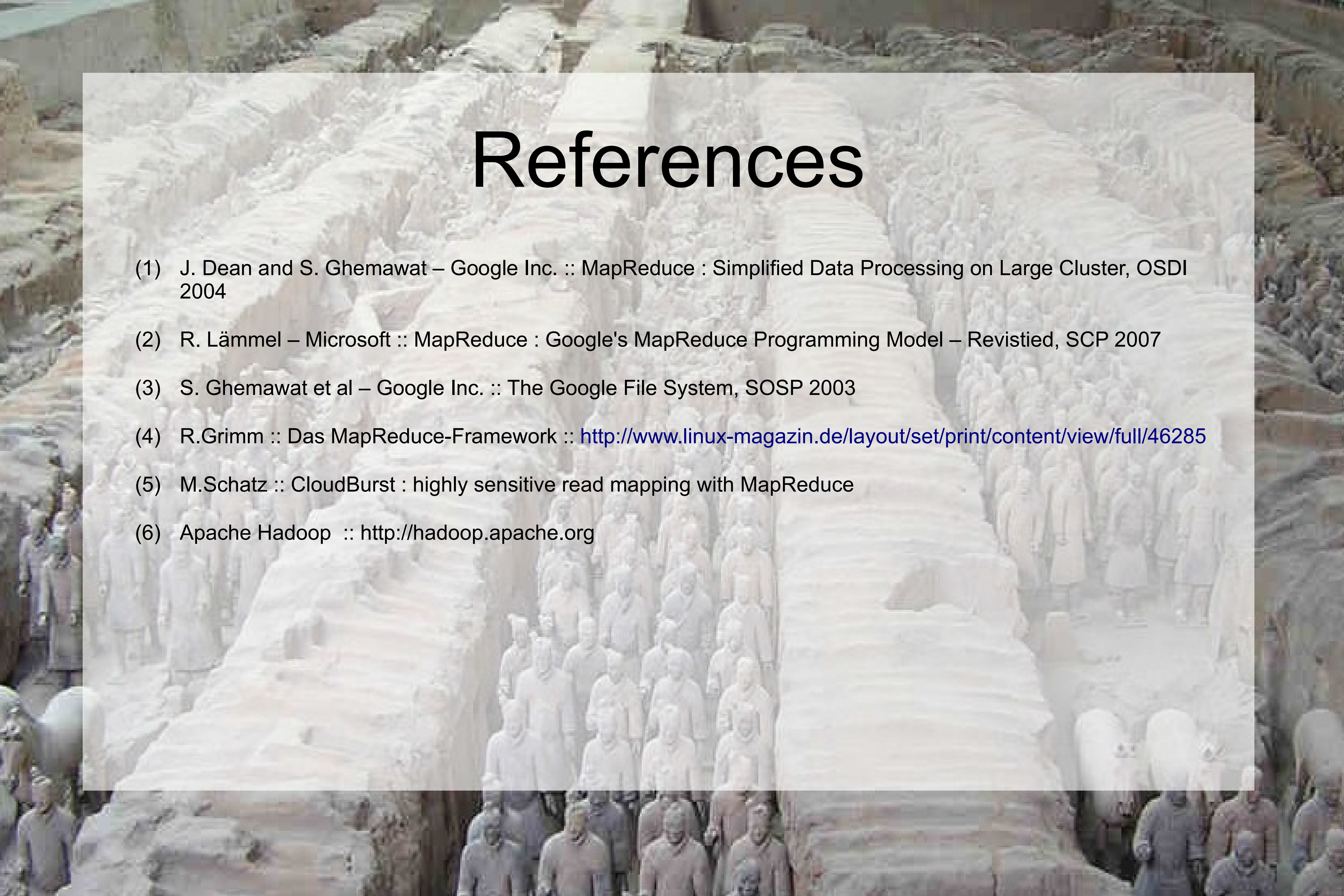
Google has told US media that "Like other responsible, innovative companies, Google files patent applications on a variety of technologies it develops. While we do not comment about the use of this, or any part of our portfolio, we feel that our behaviour to date has been in line with our corporate values and priorities". Given that Google has not pursued patent infringements and appear to have been building a defensive patent portfolio, it is believed by some that Google are ensuring that it is not possible for a patent troll to obtain a similar patent and use it against Google and others.

(djwm)

http://www.h-online.com/open/news/item/Google-patents-Map-Reduce-908602.html

End

Thanks for your attention!



References

(1) J. Dean and S. Ghemawat – Google Inc. :: MapReduce : Simplified Data Processing on Large Cluster, OSDI 2004

(2) R. Lämmel – Microsoft :: MapReduce : Google's MapReduce Programming Model – Revistied, SCP 2007

(3) S. Ghemawat et al – Google Inc. :: The Google File System, SOSP 2003

(4) R.Grimm :: Das MapReduce-Framework :: http://www.linux-magazin.de/layout/set/print/content/view/full/46285

(5) M.Schatz :: CloudBurst : highly sensitive read mapping with MapReduce

(6) Apache Hadoop :: http://hadoop.apache.org