Complete Faceting

code4lib, Feb. 2009

Toke Eskildsen

te@statsbiblioteket.dk

Mikkel Kamstrup Erlandsen

mke@statsbiblioteket.dk

Battle Plan

- Terminology (geek level: 1)
- History (geek level: 3)
- Data Structures (geek level: 6)
- Indexing (geek level: 4)
- Searching – or actually ”Counting” (geek level: 5)
- Scaling – or ”How we Cheat” (geek level: 3)
- Free Bacon! (geek level: ∞)

Wait, What is Summa?

Keywords

Search Engine

Designed for Libraries

Open Source (LGPL)

Integrated Search

Cold Facts

100% Java

Lucene Index(es)

Developed Since Winter 2005

In Production Since Nov. 2006

Lightning Talk Later

Terminology

Documents contain *Fields*

ti: Applied Quantum Mechanics
gen_subj: physics
subj: quantum mechanics

ti: Smooth Manifolds in Physics
gen_subj: mathematics
gen_subj: physics
subj: smooth manifolds

Terminology

Documents contain *Fields*

ti: Applied Quantum Mechanics
gen_subj: physics
subj: quantum mechanics

ti: Smooth Manifolds in Physics
gen_subj: mathematics
gen_subj: physics
subj: smooth manifolds

Facets contain *Tags*

The **title** facet

Applied Quantum Mechanics

Smooth Manifolds in Physics

The **subject** facet

physics

quantum mechanics

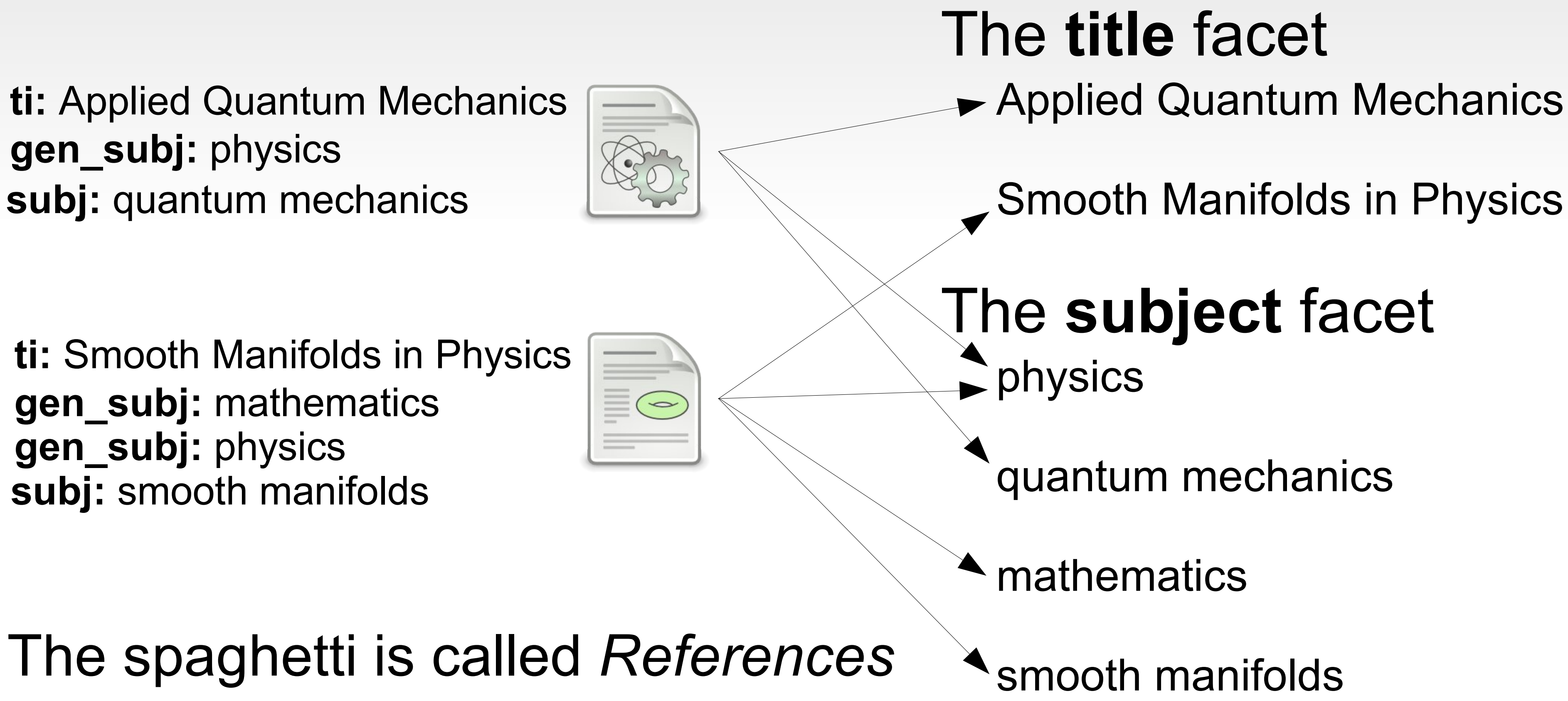
mathematics

smooth manifolds

Terminology

Documents contain *Fields*

Facets contain *Tags*



ti: Applied Quantum Mechanics
gen_subj: physics
subj: quantum mechanics

ti: Smooth Manifolds in Physics
gen_subj: mathematics
gen_subj: physics
subj: smooth manifolds

The **title** facet

- Applied Quantum Mechanics
- Smooth Manifolds in Physics

The **subject** facet

- physics
- quantum mechanics
- mathematics
- smooth manifolds

The spaghetti is called *References*

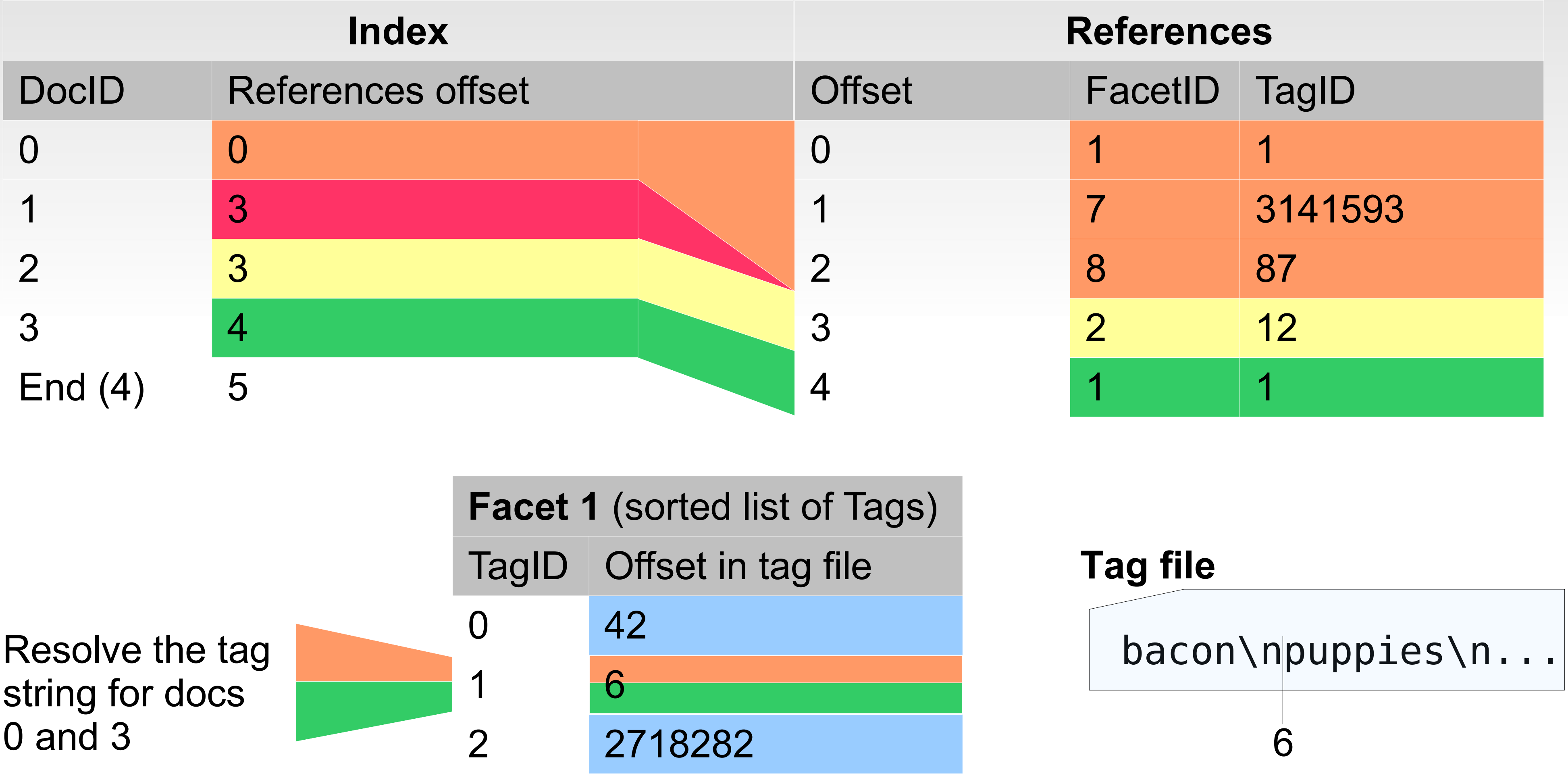
Diving In

- Iterate Lucene hits, collect field content
  - Use clean OO facet/tag structure
- Create cache map in memory
  - Collect tag counts with nice HashMap
- Logical path onwards?
  - Use field cache or similar
  - BitSets

Stop! What Do We Want?

- Scale – Up and Down
- Iterative Updates
- Decoupling from Text Search Engine

Facet Mapping



Index		References		
DocID	References offset	Offset	FacetID	TagID
0	0	0	1	1
1	3	1	7	3141593
2	3	2	8	87
3	4	3	2	12
End (4)	5	4	1	1

Resolve the tag string for docs 0 and 3

Facet 1 (sorted list of Tags)	
TagID	Offset in tag file
0	42
1	6
2	2718282

Tag file

```
bacon\npuppies\n...
```

Persistence

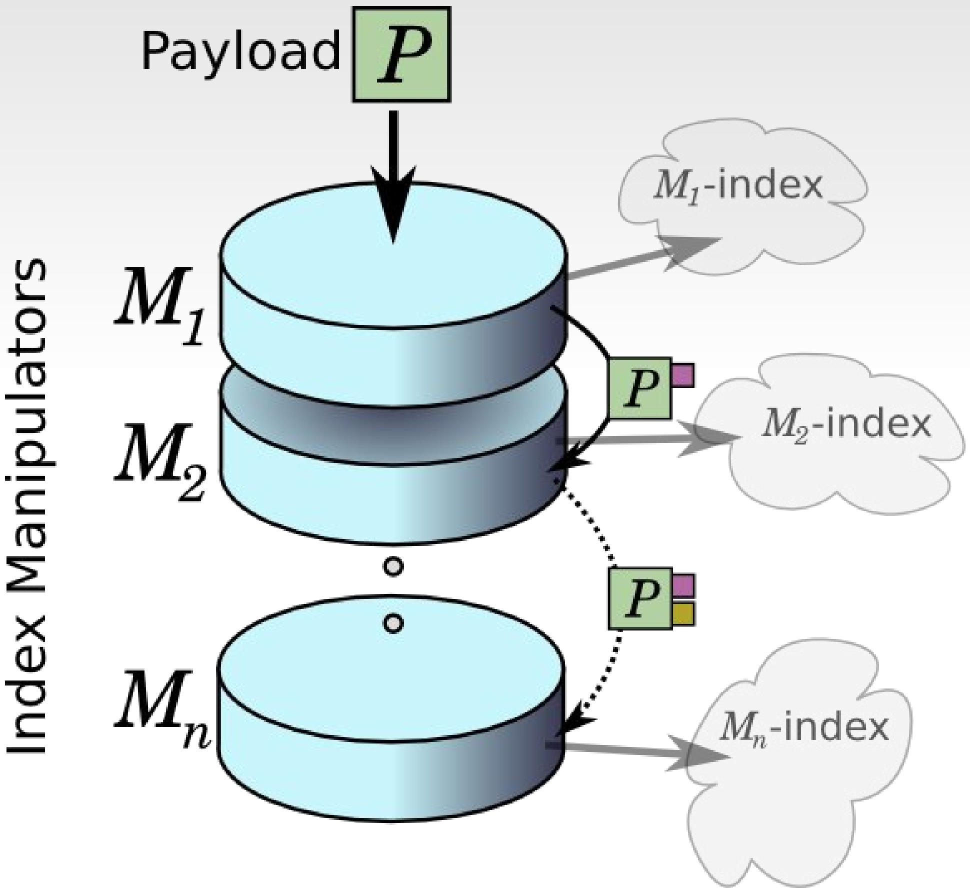
- All references are arrays
  - Just dump them directly to the file system
- Two strategies for updating tag files
  - Append tags on the fly
  - Store as a full dump at a point in time
- Two strategies for resolving tags on search
  - Get them from the file system (SSDs rules)
  - Load them fully into memory (ouch)

Persistence

- All references are arrays
    - Just dump them directly to the file system
- Two strategies for updating tag files
    - Append tags on the fly
    - Store as a full dump at a point in time
- Two strategies for resolving tags on search
    - Get them from the file system (SSDs rules)
    - Load them fully into memory (ouch)

Put those SSDs to work!

Facet structure building



Summa Manipulators:

Analyze *payload*

Write to a private *sub-index*

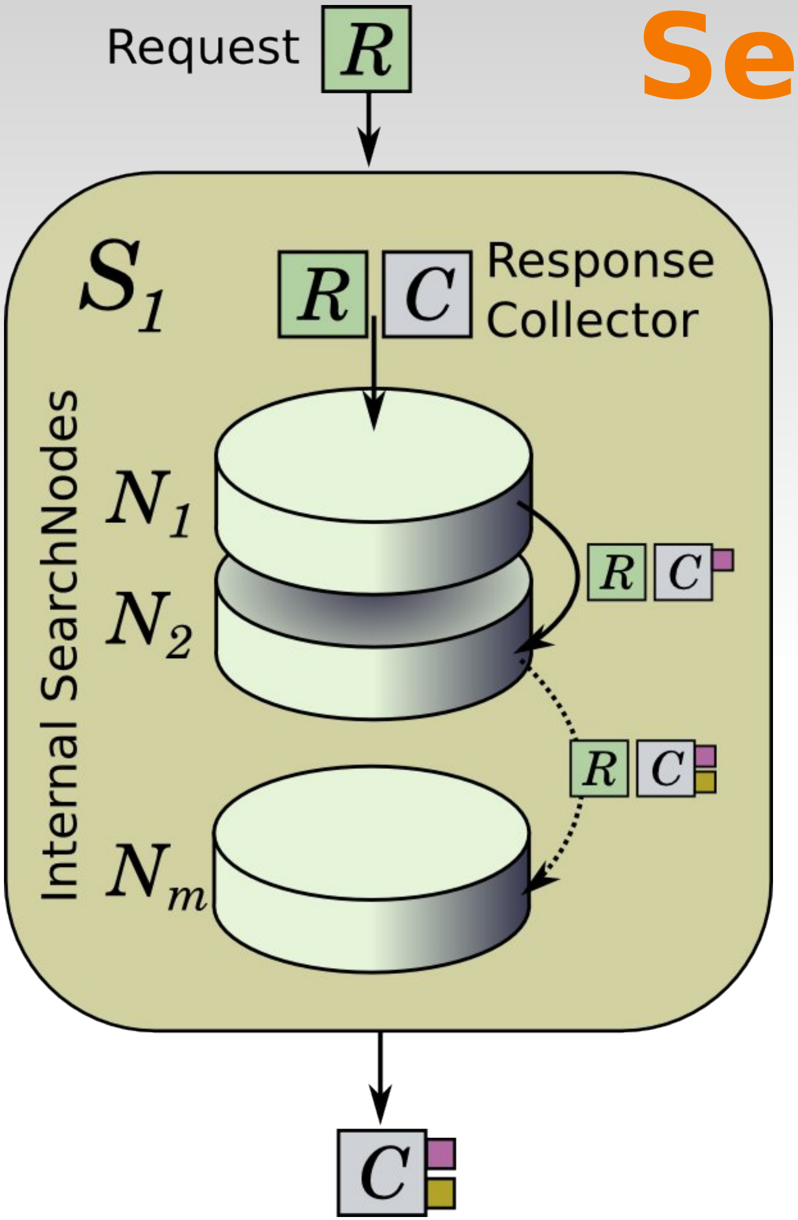
Attach additional info and pass the payload on

The Facet Manipulator:

Receives Document ID and Fields from the Document manipulator

Performs an iterative update of the facet structure

Searching



The Job of a Search Node:

Search private *sub-index*, or other private source

Add response to *collector*

The Job of the Facet Node:

Receive a *BitSet* from a previous Document Search Node

Generate Facet response, add it to the *collector*

Tag Counting

Index	
DocID	References offset
0	0
1	3
2	3
3	4
End (4)	5

References		
Offset	FacetID	TagID
0	1	1
1	7	3141593
2	8	87
3	2	12
4	1	1

Increment doc count for tag 1

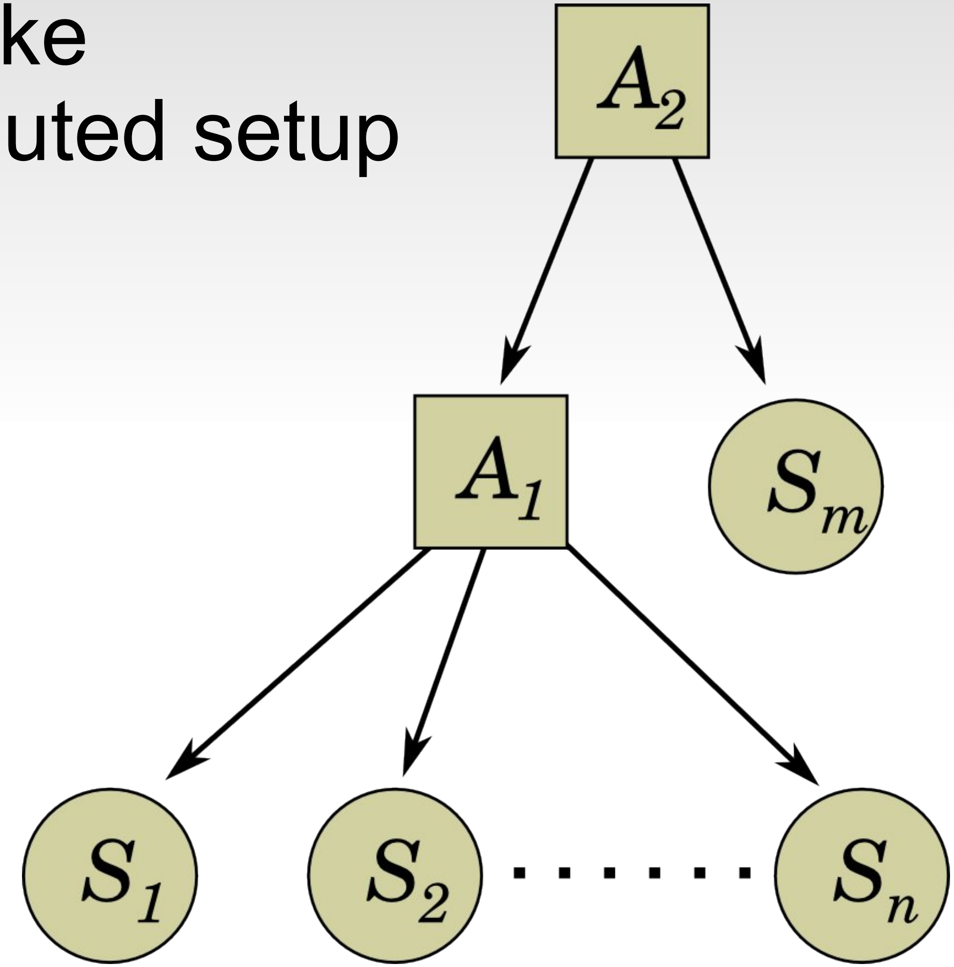
TagCounter 1 coupled to **Facet 1**	
TagID	Counter (int)
0	0
1	0 + 1 + 1
2	0

Distributed Search

Summa is equipped to take full advantage of a distributed setup

Each sub search node produces a part of the full answer

A special search node aggregates results from a set of sub search nodes



Distribution is tricky

Merge the top three tags from two nodes:

Node 1	
Tag **A**	2
Tag **B**	1
Tag **C**	1

+

Node 2	
Tag **E**	2
Tag **A**	2
Tag **D**	2

=

Result	
Tag **A**	4
Tag **D**	2
Tag **E**	2

Distribution is tricky

Merge the top three tags from two nodes:

Node 1	
Tag **A**	2
Tag **B**	1
Tag **C**	1

+

Node 2	
Tag **E**	2
Tag **A**	2
Tag **D**	2

=

Result	
Tag **A**	4
Tag **D**	2
Tag **E**	2

FAIL

Distribution is tricky

Merge the top three tags from two nodes:

~~Node 1~~	
~~Tag **A**~~	~~2~~
~~Tag **B**~~	~~1~~
~~Tag **C**~~	~~1~~

~~+~~

~~Node 2~~	
~~Tag **E**~~	~~2~~
~~Tag **A**~~	~~2~~
~~Tag **D**~~	~~2~~

~~=~~

~~Result~~	
~~Tag **A**~~	~~4~~
~~Tag **D**~~	~~2~~
~~Tag **E**~~	~~2~~

Node 1	
Tag **A**	2
Tag **B**	1
Tag **C**	1
Tag **D**	1
Tag **E**	1
Tag **F**	1

+

Node 2	
Tag **E**	2
Tag **A**	2
Tag **D**	2
Tag **B**	2
Tag **C**	1
Tag **F**	1

=

Result	
Tag **A**	4
Tag **B**	3
Tag **D**	3

Real Life in Numbers

- 10M docs, 10M tags, 100M refs, 1 machine
  - A few 1000 hits < 100 ms
  - A few 100.000 hits < 200 ms
  - 10M hits ~3 sec
- 100M docs, 1G tags, 1G refs, 3 machines
  - A few 1000 hits ~1 sec (ouch)
  - A few 100.000 hits ~1 sec
  - 10M hits < 3 sec
  - 100M hits ~15 sec

Bonus Level!

Persistent sorted Tags

- Index lookup (alphabetic listings)
- Localized range queries
- Sort without warm-up and memory overhead

Level Completed!

Dead Troll (CC) BY-NC-SA by Kim Smith (Squid@Flickr)

Questions?

wiki.statsbiblioteket.dk/summa

- Summa, Integrated Search
- Document/Field, Facet/Tag
- FieldCache, BitSet
- Iterative updates
- Lucene decoupling
- Structure
  - Persistence, Memory Overhead
- Indexing
  - WeakHashMap, MergeSort
- Tag Counting
- Distributed searching
  - Cheating
- Scalability numbers
- Collator Order