

# Scaling at Showyou

---

John Muellerleile (@jrecursive, [john@remixation.com](mailto:john@remixation.com))

September 26, 2011

# John Muellerleile

---

- Basho Technologies: Jan. 2008 - Dec. 2010
  - Riak
  - Riak Search
  - Automated research, NLP, spidering
- Consulting: 2003 - 2008
  - E-commerce, AdSense, AdWords, ...
- Infrastructure:
  - Messaging
  - Riak
  - Solr


# Agenda

---

- **Who am I?**
- **What is Showyou?**
- **The Nature of “Social Data”**
- **Showyou’s Data:** Today & Tomorrow
- **Data Management:** Technology Stack
- **Riak:** The Awesome & Sub-Awesome, Integration Patterns & Observations
- **Not Bob’s Riak:** The “**Mecha**” Backend & Query System

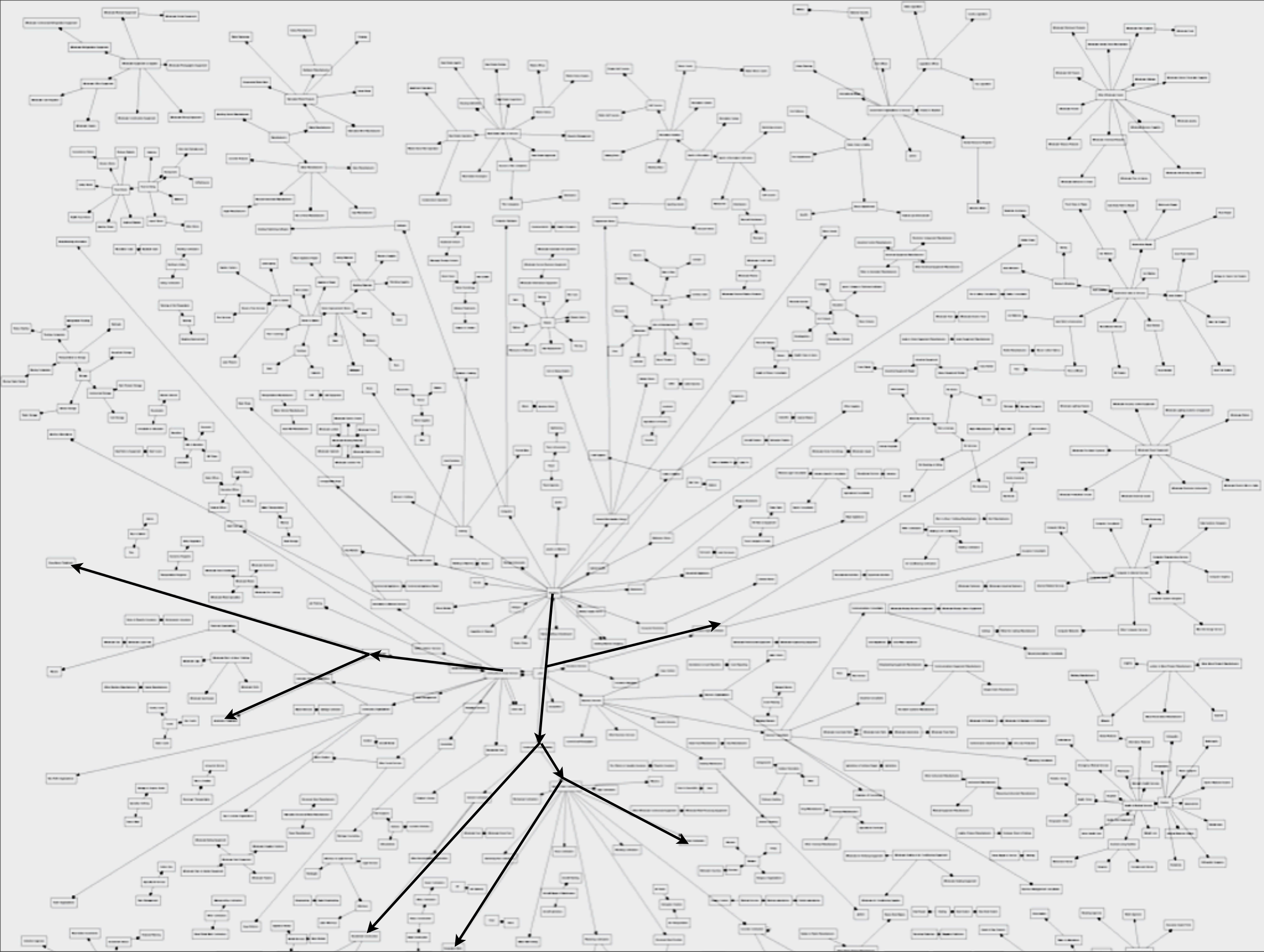
# What is Showyou?

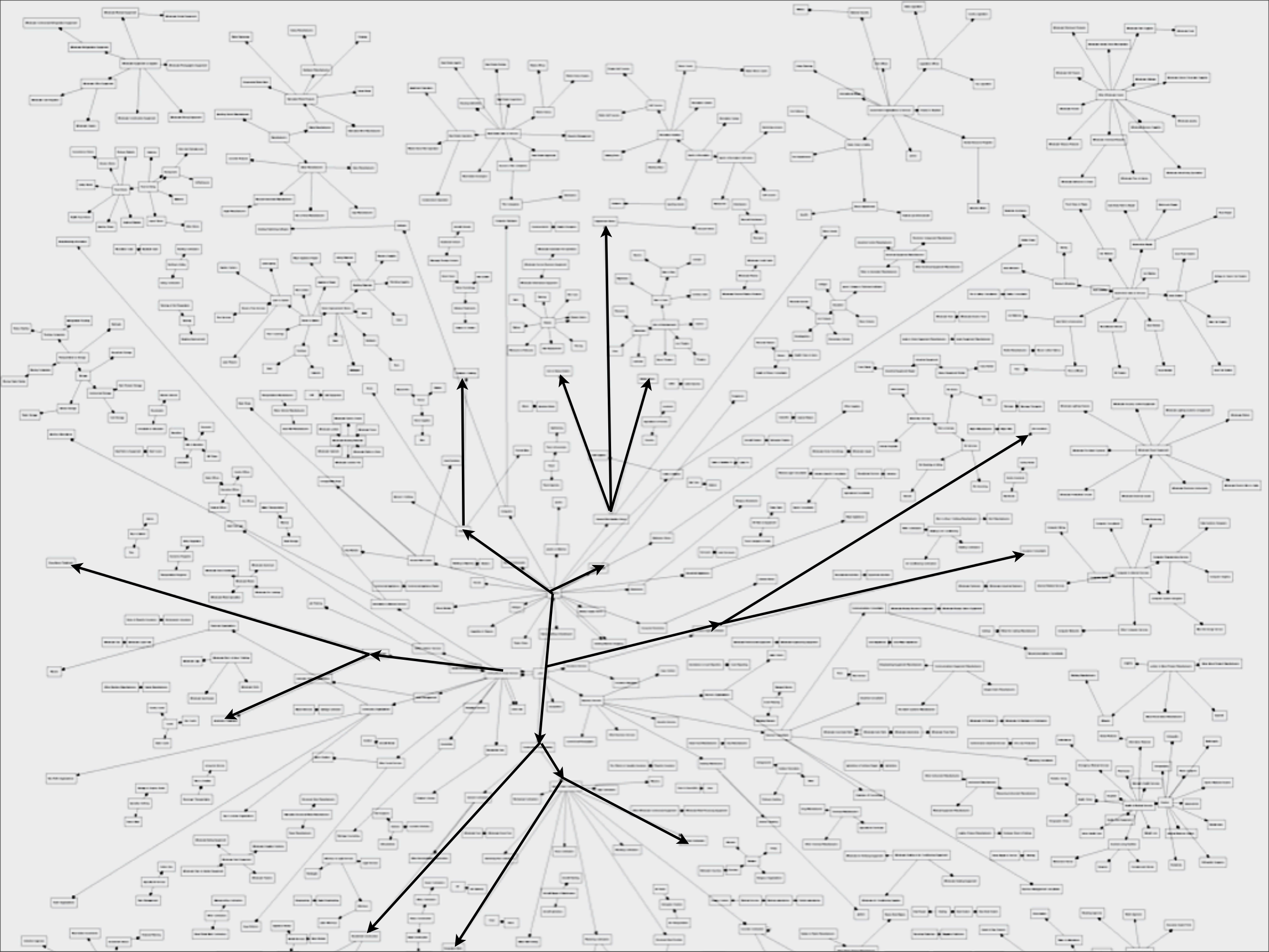




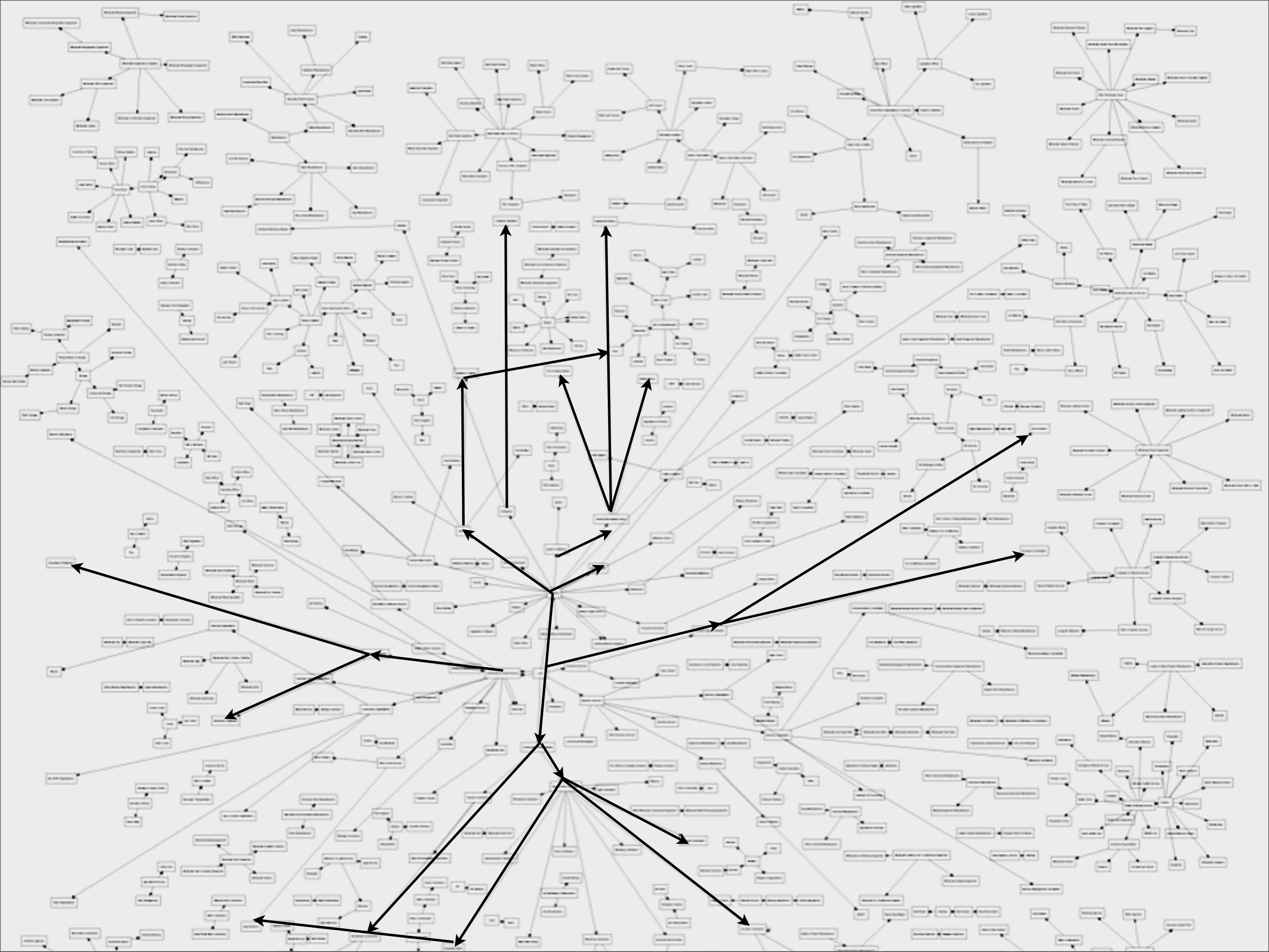
# A minute about the nature of “Social Data”



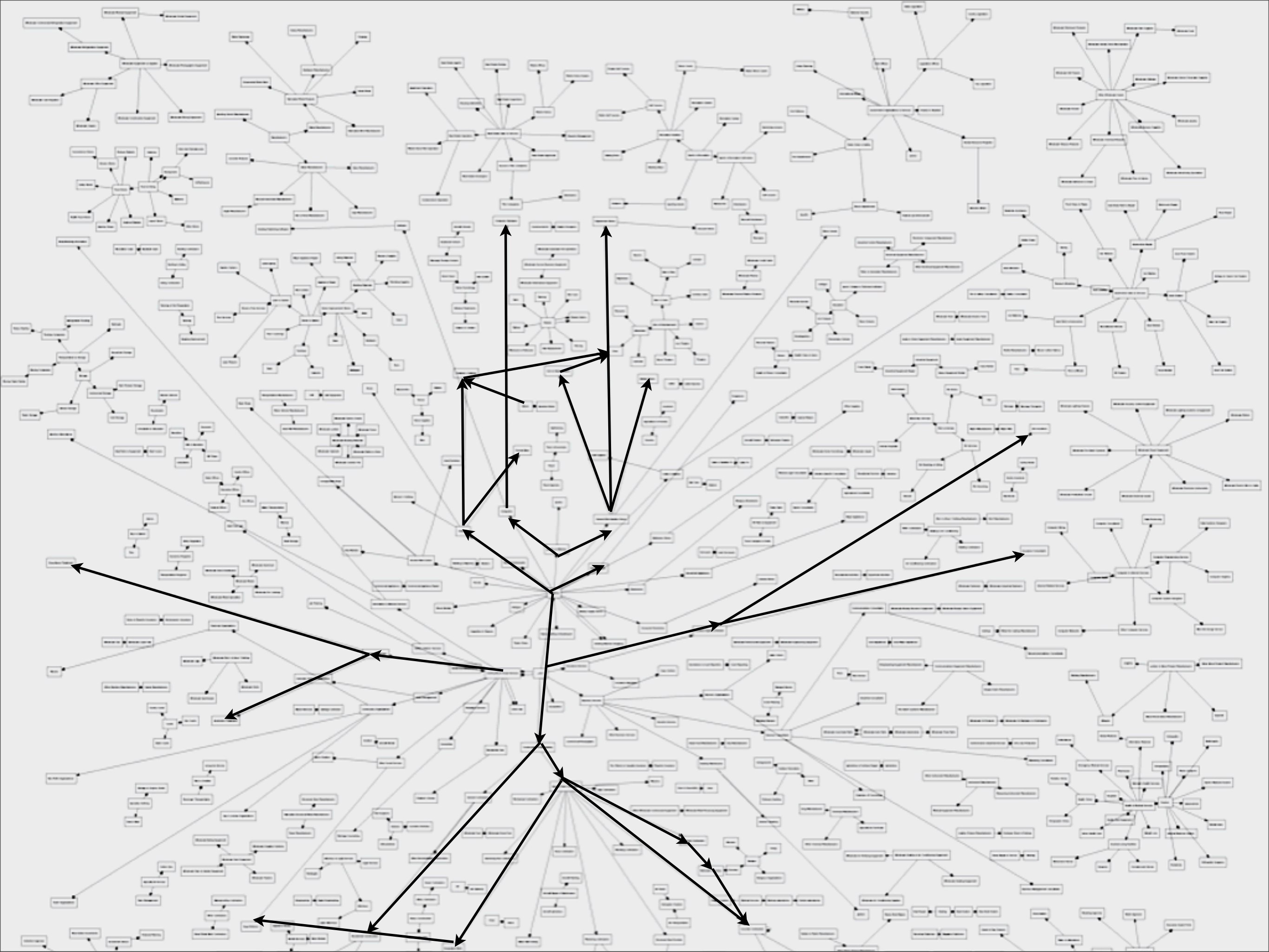


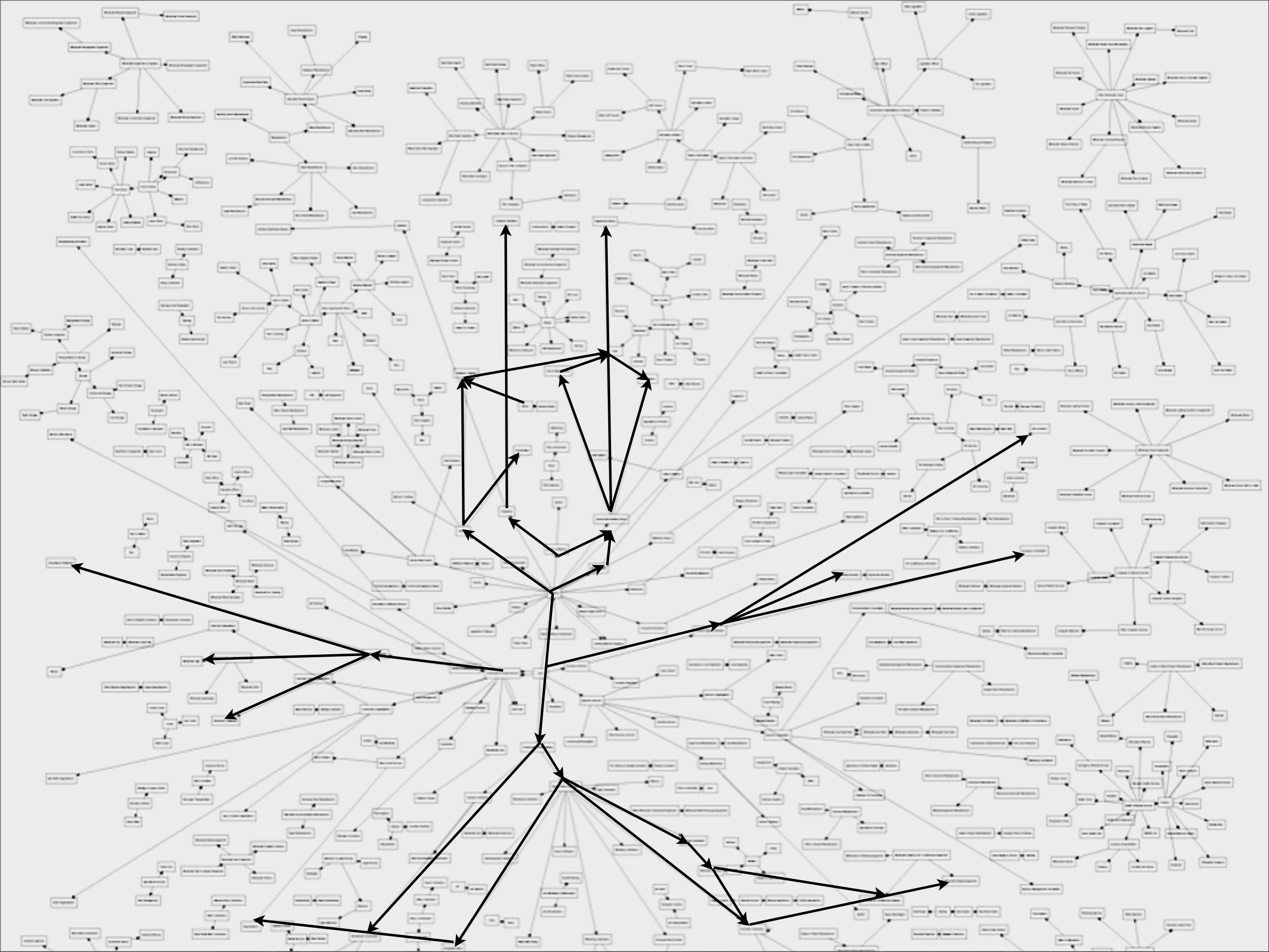




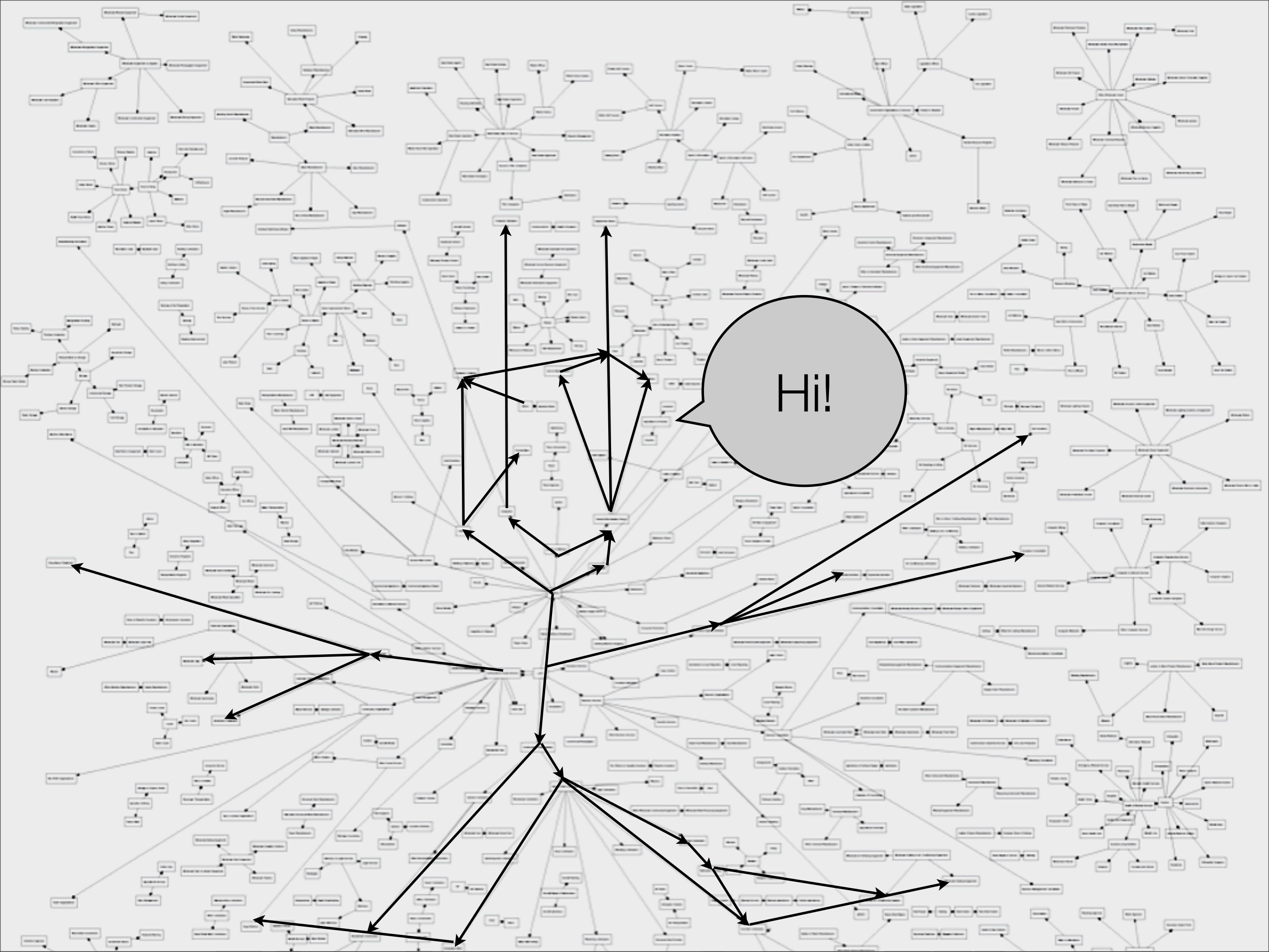














# Showyou's Data: Today

---

- **Riak** with Bitcask
- **Solr** with replication
- Often ever-growing blobs of **JSON**
- No useful way to “find” data based on anything other than compound primary keys :(
  - Primary keys crafted for specific access patterns :(
- This will not last forever

# Showyou's Data: Tomorrow

---

- **Significant data growth** per additional user
- **Find & aggregate data** about our users & their videos
- **Derive useful “signal”** from this data
  - **Better search**: disambiguation, “more like this”, performance
  - **“Collective Intelligence”**: trending, “smart collections”
  - **Spam & De-duplication**
  - & more: *#hashtags, auto-complete, statistics, usage, ...*

# Data Management: Technology Stack

---

- **Erlang/OTP**
  - Riak, Bitcask
- **Java**
  - JInterface
  - Hazelcast
  - Solr



# Riak: The Awesome Parts

---

- **By far the best operational story in its class**
- **Shared-nothing**
- **No single point of failure**
- **Masterless multi-site replication**
- **Support** via EnterpriseDS Startup Program
- I helped write it & turned it inside-out to design Riak Search while working at Basho -- *this helps*

# Riak: The Sub-Awesome Parts

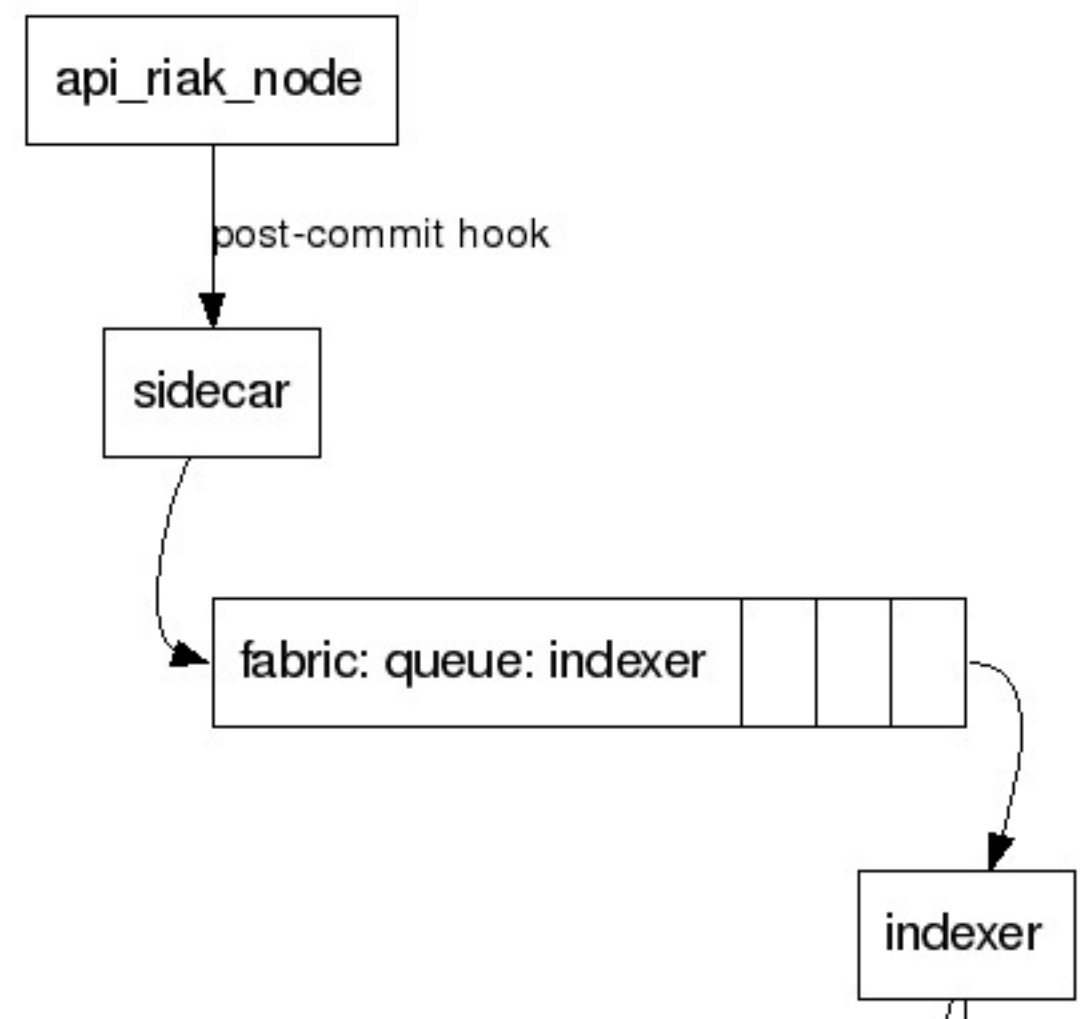
---

- Riak Search as it exists today does not perform well for us & lacks features
- Bitcask keeps **all keys in memory**
- **Listing keys for a bucket will take your cluster down**
- **Map/Reduce is virtually useless** (for us) other than as “multi-get”
- Pre-1.0 cluster membership changes are “at your peril”
- **Usable/useful built-in monitoring is non-existent**
- *If you are not intimately familiar with Riak, it's very hard to debug!*

# Riak: Integration Patterns

---

- **api\_riak\_node**: “main” Riak cluster node
- **sidecar**: post-commit talks to local Java-based Erlang node using JInterface
- **fabric**: distributed data structures & utilities via Hazelcast - very similar in spirit & implementation as Riak!
- **indexer**: pull from fabric queue & index record in Solr
- **Identical deployment on every node**

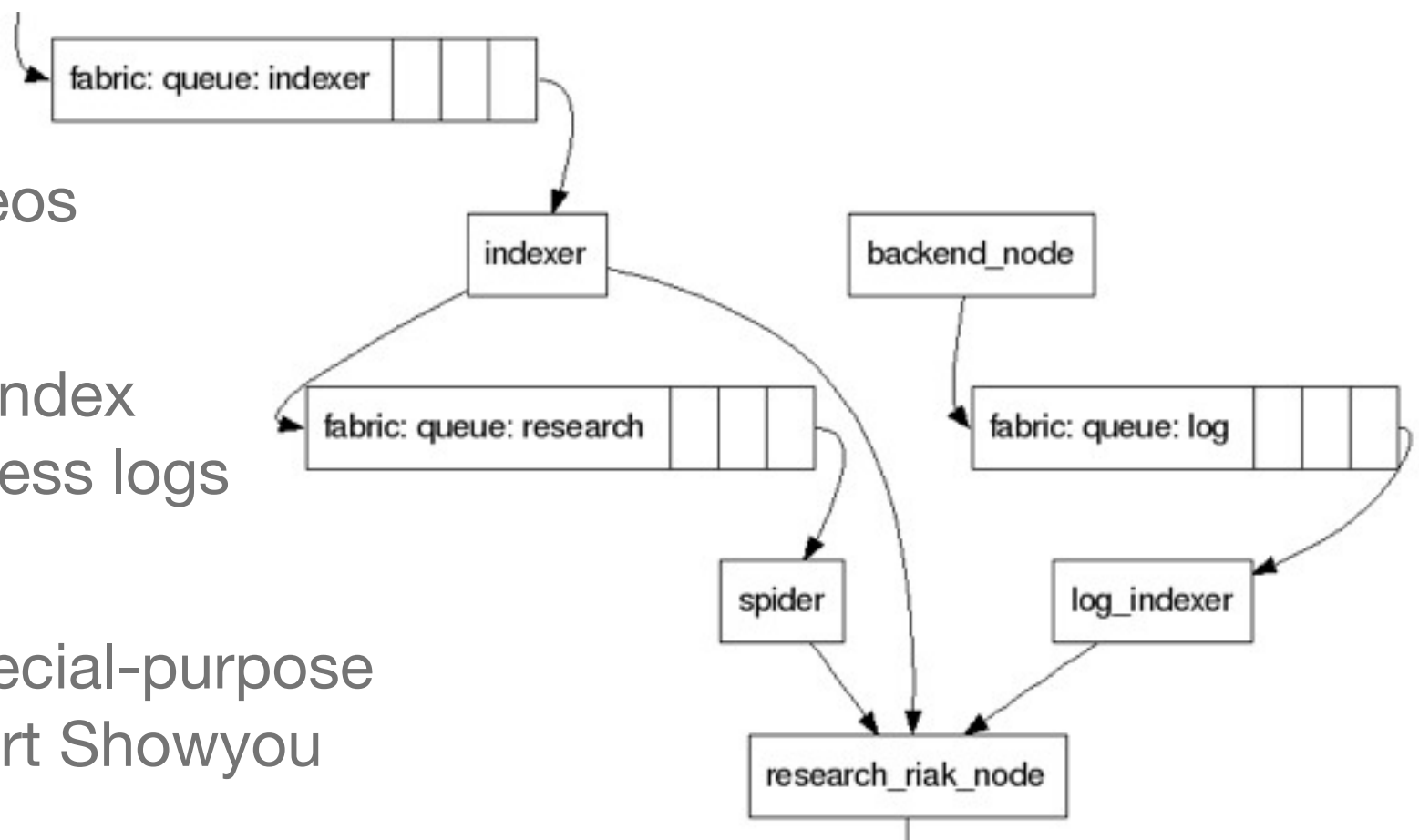




# Riak: Integration Patterns: The Big Picture

---

- **backend\_node**: nginx, redis; logs
- **spider**: finds, extracts, stores & indexes further information on users & videos
- **log\_indexer**: aggregate & index interesting parts of our access logs
- **research\_riak\_node**: a special-purpose Riak cluster node to support Showyou data “Tomorrow”



# Riak: Integration Patterns: Observations

---

- Riak post-commit hooks
- Why not pre-commit hooks?
- **Riak as a “virtual memory”**
- **Post-commit hooks as “change events”**
- **Wishlist: pre- & post-delete hook** (I realize this is tricky - do it anyway)
- **Wishlist: pre- & post-create hook** (Less tricky - do it anyway)

# Riak: Integration Patterns: Search

---

- Monolithic replicated Solr won't last forever & sharding is a faceted multi-value “*shitshow*” [1]
- We're doing fine, for now, with lots of RAM, SSDs, etc. but...
- I don't want to find out “*the hard way*” where the joyride ends and the hellride starts [2]
- **Clearly the answer is to kill every bird in nearby airspace by writing a Riak storage backend and integrated query mechanism!** [2,3,4]

[1] Cliff Moon suggested the use of this word (for emphasis)

[2] It's okay, I have done this several dozen times

[3] Yes, really: BDB, BDBJE, Innostore, sqlite, hsql, mysql, postgres, etc.

[4] This is not a pride thing: it was a hard, lonely, unforgiving road

# Not Bob's Riak: A Moment of Weakness

---

“My way of joking is to tell the truth;  
it's the funniest joke in the world”  
George Bernard Shaw



# Not Bob's Riak: Introducing "Mecha"

- Bob?

## Fault-tolerance

by @jrecursive



# Mecha: Goals

---

- **Birds to kill:**
  - **Tight, purposed Riak integration**
  - **Efficient & feature-complete indexing**
  - **Fast sequential & range object access**
  - **Flexible distributed query mechanism**
  - **Query parallelism** where appropriate

# Mecha: What Stays The Same

---

- **Works with “stock” (unmodified) Riak 1.0 (pre & release)**
- Little/no difference from the “Riak side” -- everything works “as it should”
- **Differences:**
  - All objects you “put” into Riak must be JSON objects (this will change by release to respect content type)
  - Any fields without a specific “indexed field type” (e.g., \_t, \_s, \_dt, etc.) are simply stored along with the rest of the fields (i.e. “stored field”)

# Mecha: At a glance

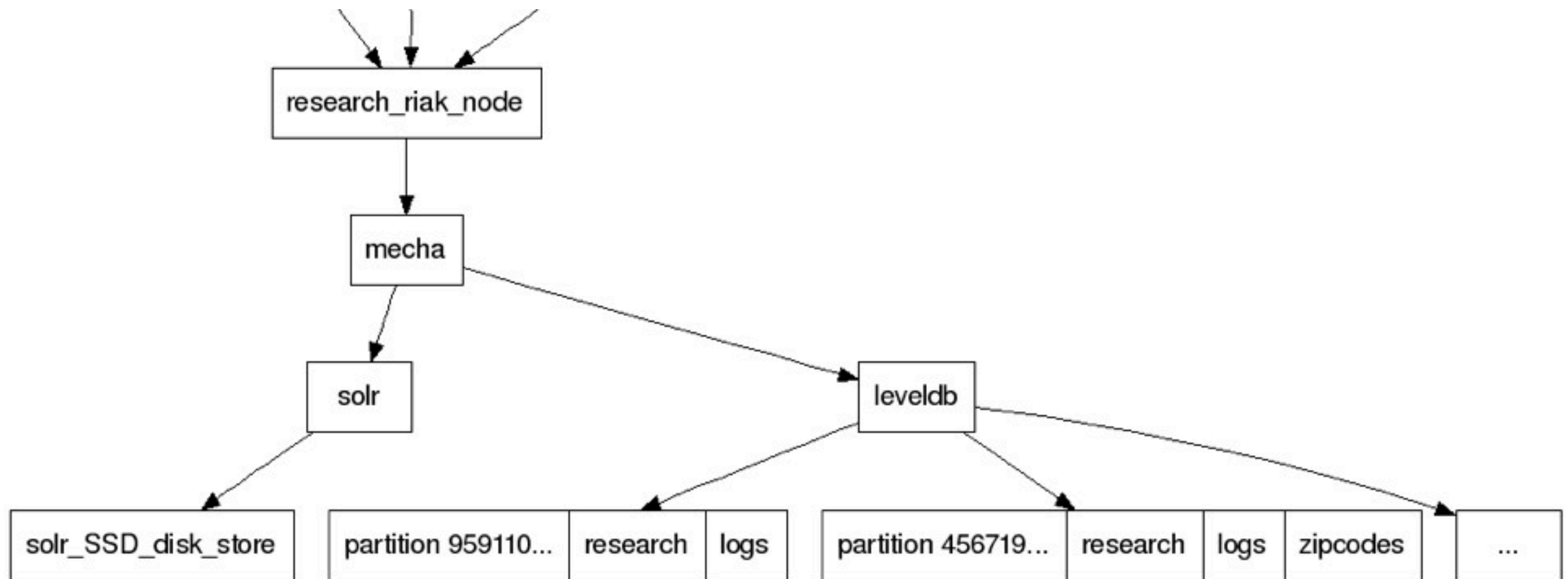
---

- Written in Java, uses many, many third-party libraries: LevelDB (JNI), Solr, JInterface, Jetlang, Netty, Protobufs, Commons, ...
- Riak backend module written in Erlang; beaten into submission for reliable interaction with JInterface-driven Java node
- LevelDB instance *per partition, per bucket*; “ulimit -n 90000” :)
- One Solr instance per node (covers all partitions, buckets)
- Objects stored as Riak Objects *in JSON form* in LevelDB
- Standardized schema covering common data types using name *suffixes*



# Mecha: Riak Integration

---



# Mecha: Index Field Types

---

- **Supported index field types** (by suffix):
- **\_t, \_tt** - **full-text** (optionally w/ term vectors, ...)
- **\_s, \_s\_mv** - exact **string**, **multi-value** exact string
- **\_i, \_l, \_d, \_f** - trie-based **integer**, **long**, **double**, **float**
- **\_dt** - trie-based **date** (YYYY-MM-DDTHH:MM:SS)
- **\_b** - **boolean**
- **\_xy, \_ll, \_geo** - **point**, **lat/lon** & **geohash**

# Mecha: Example Object

---

```
{ "content_t": "NoSQL is a ghetto",  
  "lol_count_i": 47,  
  "lol_dt": "2009-04-13T07:01:43.000Z"  
  "rating_f": 4.111111164093018,  
  "tags_s_mv": [  
    "funny",  
    "lol",  
    "nosql",  
    "ghetto"]}
```

# Mecha: Fast Sequential & Range Object Access

---

- **Every bucket gets own instance of LevelDB per partition**
- **No “multiplexing”** buckets or partitions per LevelDB instance
- **Keys are literal**, no encoded Erlang terms; simple ranges, smaller values
- Values stored as **JSON-ized Riak objects** (why? you’ll see)
- LevelDB JNI binaries shipped with **Snappy compression built-in**



# Mecha: Flexible Distributed Querying

---

- **Exact, prefix, suffix, & wildcard filtering** on multiple index fields
- **Ultra-fast list\_keys, list\_bucket, list\_buckets replacements**
- Equally **fast bucket count** operation
- Ridiculous **range query performance** (any trie- type; sane datetime functions)
- **Faceting, group counts**
- **Spatial** (bounding box/bowl, geofilt, Haversine, distance faceting)

# Mecha: Query Parallelism

```
mecha@x.x.40.78> f research location_s last_modified:[NOW-1HOUR TO NOW] -location_s:undefined
```

```
http://x.x.40.76:7331/solr/select/?q=*
%3A*&version=2.2&wt=json&start=0&rows=0&facet=true&facet.limit=10&facet.field=location_s&facet.mincount=1&fq=buc
ket%3Aresearch+partition%3A%28639406966332270026714112114313373821099470487552+OR
+274031556999544297163190906134303066185487351808+OR+1187470080331358621040493926581979953470445191168+OR
+456719261665907161930651510223838443642478919680+OR+1370157704997721485815954530671515330927
+91343852333181432387730302044767688720495783936+OR+1004702375664995756265033224924445760134
+822094670998632891489572718402909198556462055424%29+last_modified%3A%5BNOW-1HOUR+TO+NOW%5D+-
%3Aundefined
http://x.x.76.2:7331/solr/select/?q=*
%3A*&version=2.2&wt=json&start=0&rows=0&facet=true&facet.limit=10&facet.field=location_s&facet.mincount=1&fq=buc
ket%3Aresearch+partition%3A%28365375409332725729550921208179070754913983135744+OR+0+OR
+913438523331814323877303020447676887204957839360+OR+182687704666362864775460604089535377456991567872+OR
+1278813932664540053428224228626747642198940975104+OR+1096126227998177188652763624537212264741949407232+OR
+540063113999000594326301812260606132370974703616+OR
+730750818665451459101842416358141509827966271488%29+last_modified%3A%5BNOW-1HOUR+TO+NOW%5D+-location_s
%3Aundefined
```

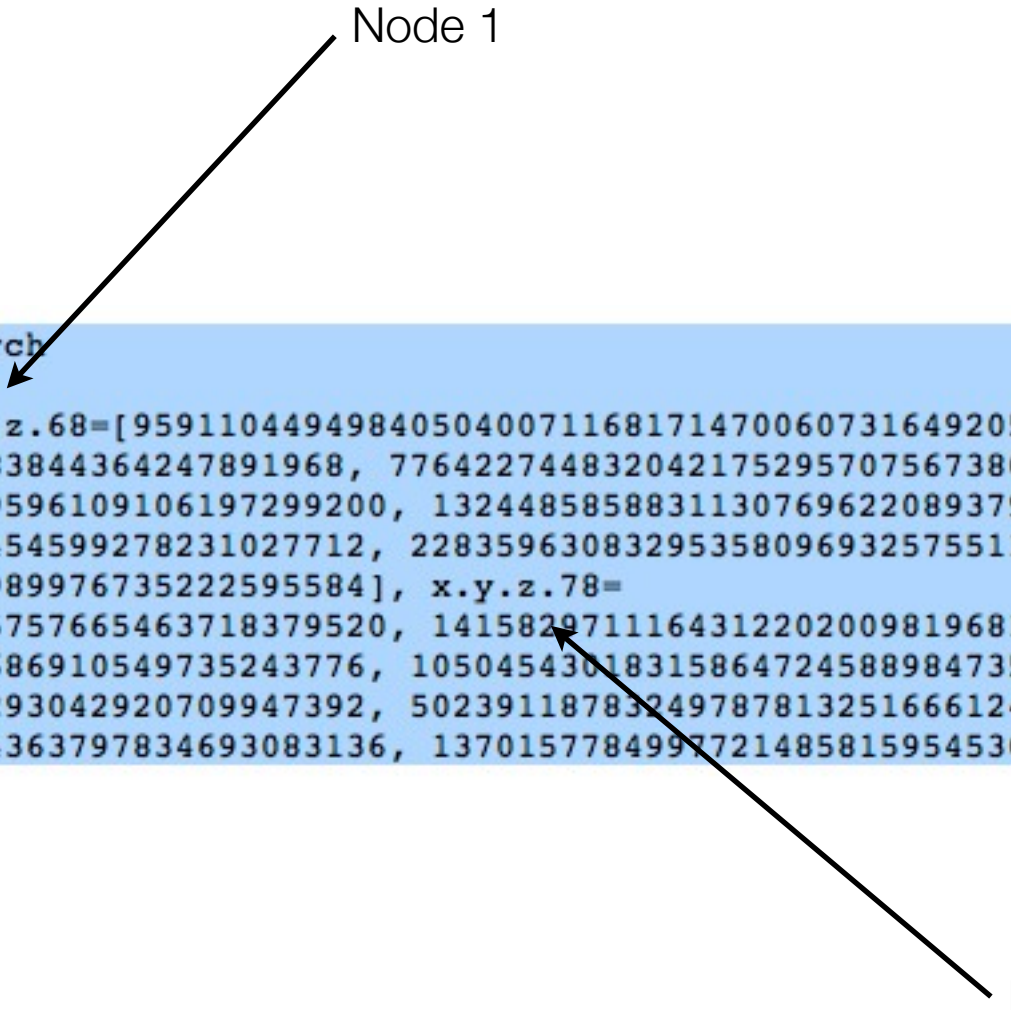


```
INFO: Los Angeles, CA: 6
INFO: Berlin: 5
INFO: EV Filmmaker Program: 30
INFO: Madrid: 5
INFO: NL Hilversum: 8
INFO: Washington, DC: 4
INFO: London: 28
INFO: paris: 6
INFO: Manchester, NH: 4
INFO: Washington DC: 4
INFO: New York, NY: 51
INFO: Cincinnati, OH: 6
INFO: Paris: 4
INFO: Austin, TX: 3
INFO: Great Falls, VA: 18
INFO: Istanbul, Turkey: 6
```

# Mecha: Coverage

---

- Coverage is the set of nodes and respectively owned partitions you must process to cover all objects in a given bucket.



```
mecha@x.y.z.78> coverage research
INFO: coverage: research: {x.y.z.68=[959110449498405040071168171470060731649205731328,
45671926166590716193865151022383844364247891968, 776422744832042175295707567380525354192214163456,
1141798154164767904846628775559596109106197299200, 1324485858831130769622089379649131486563188867072,
411047335499316445744786359201454599278231027712, 228359630832953580969325755111919221821239459840,
593735040165679310520246963290989976735222595584], x.y.z.78=
[685078892498860742907977265335757665463718379520, 1415829711164312202009819681693899175291684651008,
319703483166135013357056057156686910549735243776, 1050454301831586472458898473514828420377701515264,
867766597165223607683437869425293042920709947392, 502391187832497878132516661246222288006726811648,
1233142006497949337234359077604363797834693083136, 137015778499772148581595453067151533092743675904]]}
```

(for n=2)

Node 2



# Mecha: Examples: Count & Faceting

---

- Count the number of records in the "research" bucket modified within the last 10 seconds

```
mecha@x.y.z.78> c research last_modified:[NOW-10SECOND TO NOW]
```

```
INFO: count: 38  
INFO: 7ms elapsed
```

- Top search terms by count for the last hour

```
mecha@x.y.z.78> f logs q_q_s last_modified:[NOW-1HOUR TO NOW]
```

```
INFO: zac brown: 52  
INFO: peter capusotto: 42  
INFO: hot girl: 62  
INFO: icloud: 22  
INFO: ces 2011 ipad: 292  
INFO: sir ken robinson: 84  
INFO: ces 2011: 399  
INFO: axe: 98  
INFO: photography: 20  
INFO: hot sex webcam: 35  
[... cut ...]  
INFO: 11ms elapsed
```

# Mecha: Examples: Multi-value String Fields

---

- See which field values occur with other values, and how often (using a multi-value string field)

```
mecha@x.y.z.78> f research tags_s_mv tags_s_mv:dubstep
INFO: music: 668
INFO: new: 97
INFO: bass: 299
INFO: 2011: 112
INFO: drum: 170
INFO: remix: 260
INFO: dub: 169
INFO: dance: 98
INFO: entertainment: 9
INFO: electronic: 137
```



# Mecha: Next Steps

---

- Code cleanup, test & polish current functionality
- Sane build & deployment (yay, Maven)
- Simplify configuration
- Embed Solr (currently running standalone for debugging)
- Extend & improve Solr “standard configuration”
- “Out of Band” Map/reduce with “direct bucket-level” LevelDB instance
- After that? *Join operators, ... :)*

# Mecha: Availability

---

- Right now it is a complex system
- It will be worth waiting for tighter integration & polish
- *This is me not answering your question :)*
- As soon as responsible, sane, & possible :)

# Thank you!

---

- Basho Technologies, especially Andy Gross (@argv0) & Kelly McLaughlin (@\_klm) specifically for help with Riak 1.0 changes
- Of course, without Erlang/OTP, LevelDB, Solr, Java, JInterface, and a host of other open source projects, this would have never even gotten started -- thank you.
- Last but not least, thank you to my Showyou teammates for encouragement & support!
- Contact:  
John Muellerleile / <http://twitter.com/jrecursive>  
[john@remixation.com](mailto:john@remixation.com), [jmuellerleile@gmail.com](mailto:jmuellerleile@gmail.com)  
<http://github.com/jrecursive>