Lucene Server

From Erlang to Java and Back Again

Online at: http://prezi.com/user/inaka/

Lucene Server

From Erlang to Java and Back Again

Online at: http://prezi.com/user/inaka/


Hello World!!
About Me
• Programmer since the age of 10
• Coded in Basic, .NET, Java, JS, ...
• Erlang programmer for the last 5 years
• Director of Engineering at Inaka
About Inaka
• We're based in Argentina
• We do Erlang consulting
• We build end-to-end applications with Erlang, Android and iOS components
• We develop highly-concurrent servers for applications like Whisper or TigerText
About TigerText
• Provides secure communication for mobile devices
• Heavily used by health-care companies
• Also provides the underlying communication platform for other apps
• All servers written in Erlang

About Me

- Programmer since the age of 10 
- Coded in Basic, .NET, Java, JS, …
- Erlang programmer for the last 5 years
- Director of Engineering at Inaka

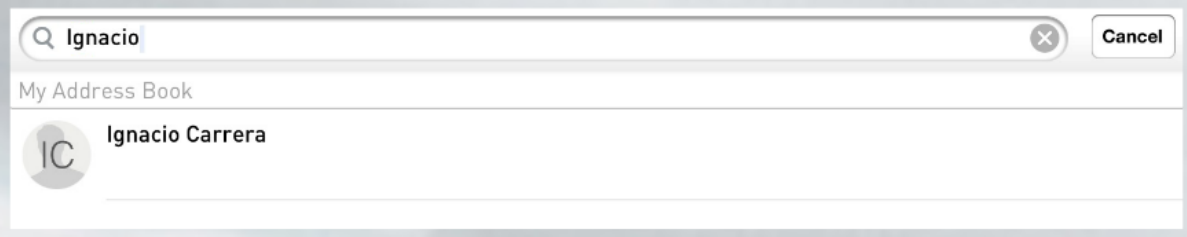

About Inaka
• We're based in Argentina
• We do Erlang consulting
• We build end-to-end applications with Erlang, Android and iOS components
• We develop highly-concurrent servers for applications like Whisper or TigerText


About TigerText
• Provides secure communication for mobile devices
• Heavily used by health-care companies
• Also provides the underlying communication platform for other apps
• All servers written in Erlang

The Story

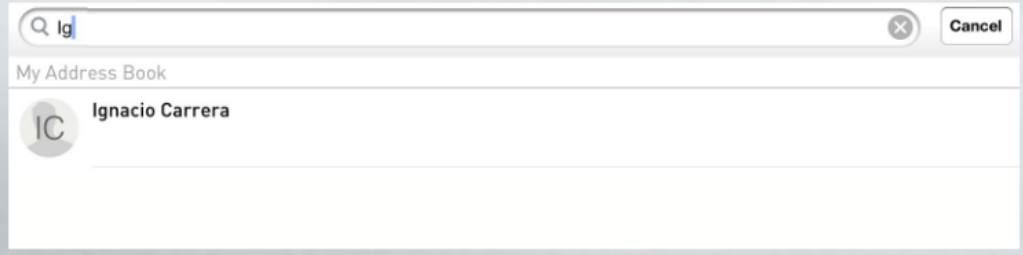

The Scenario
Functional Requests
Technical Requests
Lucene Core
It's written in JAVA
JInterface
It comes with Erlang!
Lucene Server

The Scenario



We had a very basic search field for users…

Functional Requests



We wanted to add the following features:

- Partial Matching
- Autocomplete
- Advanced Search (by field, location, etc.)
- Ranked Results

Technical Requests

We also wanted our solution to be:

- Implemented in Erlang
- Included in our existing servers as a rebar dependency
- Fast and scalable

Lucene Core

A high-performance, scalable, full-featured text search engine library with powerful, accurate and efficient search algorithms

...but...

It's written in JAVA

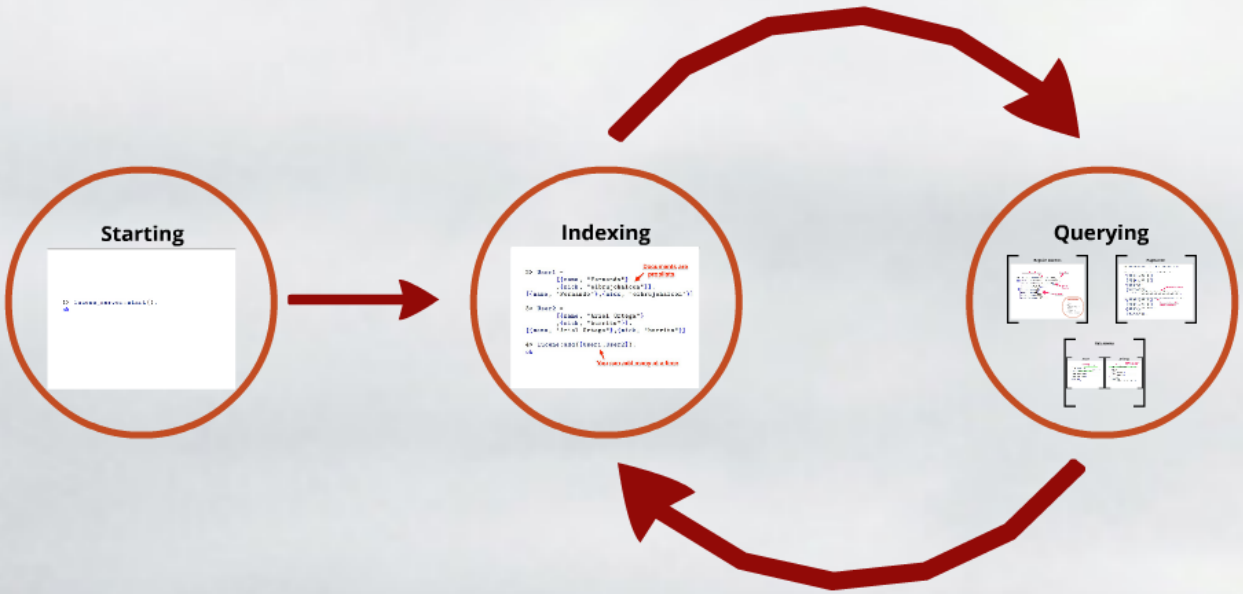

JInterface
A Java library that let's you create "nodes" in Java
and provides the means to communicate them
with regular Erlang nodes
…and…
It comes with Erlang!

Lucene Server

An OTP compliant application that provides indexing and querying access to a lucene backend without any JAVA knowledge requirement for its users


How it works
Lucene Server
How it's done

How it works



Starting

```
1> lucene_server:start().
ok
```

Indexing

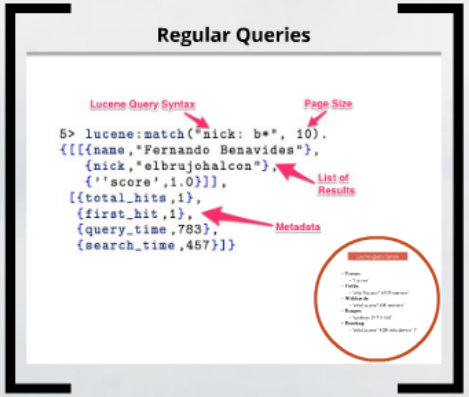
```
2> User1 =
        [{name, "Fernando"}
        ,{nick, "elbrujohalcon"}].
[{name, "Fernando"},{nick, "elbrujohalcon"}]

3> User2 =
        [{name, "Ariel Ortega"}
        ,{nick, "burrito"}].
[{name, "Ariel Ortega"},{nick, "burrito"}]

4> lucene:add([User1,User2]).
ok
```

Documents are proplists

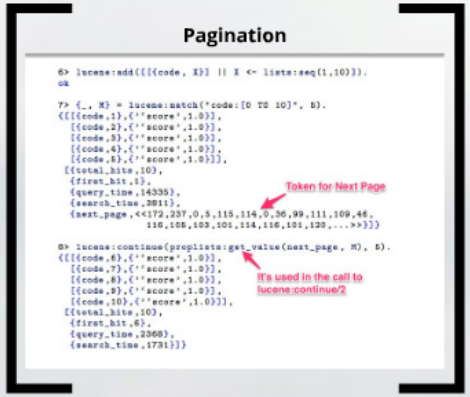
You can add many at a time

Querying

Regular Queries



Pagination



Extensions

.near

.erlang

Regular Queries

Lucene Query Syntax

Page Size

```
5> lucene:match("nick: b*", 10).
{[[{name,"Fernando Benavides"},
   {nick,"elbrujohalcon"},
   {''score',1.0}]],
 [{total_hits,1},
  {first_hit,1},
  {query_time,783},
  {search_time,457}]}
```

List of Results

Metadata

Lucene Query Syntax

- **Terms:**
  - 'Lucene'
- **Fields:**
  - 'title:"Lucene" AND text:test'
- **Wildcards:**
  - 'title:Lucene* OR text:te?t'
- **Ranges:**
  - 'ranking: [5 TO 10]'
- **Boosting:**
  - 'title:Lucene^4 OR title:Server^1'

Lucene Query Syntax

- **Terms:**
  - 'Lucene'
- **Fields:**
  - 'title:"Lucene" AND text:test'
- **Wildcards:**
  - 'title:Lucene* OR text:te?t'
- **Ranges:**
  - 'ranking: [5 TO 10]'
- **Boosting:**
  - 'title:Lucene^4 OR title:Server^1'

Pagination

```
6> lucene:add([[{code, X}] || X <- lists:seq(1,10)]).
ok

7> {_, M} = lucene:match("code:[0 TO 10]", 5).
{[[{code,1},{''score',1.0}],
  [{code,2},{''score',1.0}],
  [{code,3},{''score',1.0}],
  [{code,4},{''score',1.0}],
  [{code,5},{''score',1.0}]],
 [{total_hits,10},
  {first_hit,1},
  {query_time,14335},
  {search_time,3811},
  {next_page,<<172,237,0,5,115,114,0,36,99,111,109,46,
               116,105,103,101,114,116,101,120,...>>}]}

8> lucene:continue(proplists:get_value(next_page, M), 5).
{[[{code,6},{''score',1.0}],
  [{code,7},{''score',1.0}],
  [{code,8},{''score',1.0}],
  [{code,9},{''score',1.0}],
  [{code,10},{''score',1.0}]],
 [{total_hits,10},
  {first_hit,6},
  {query_time,2368},
  {search_time,1731}]}
```

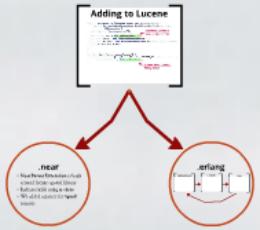
Token for Next Page

It's used in the call to lucene:continue/2

How it's done

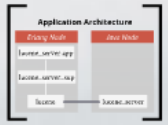
An OTP Application

Extensions



A gen_server in Java

An OTP Application



The Goals

- rebar compatible
- application:start/stop compatible
- Start and stop the Java node automatically
- Java node should be supervised

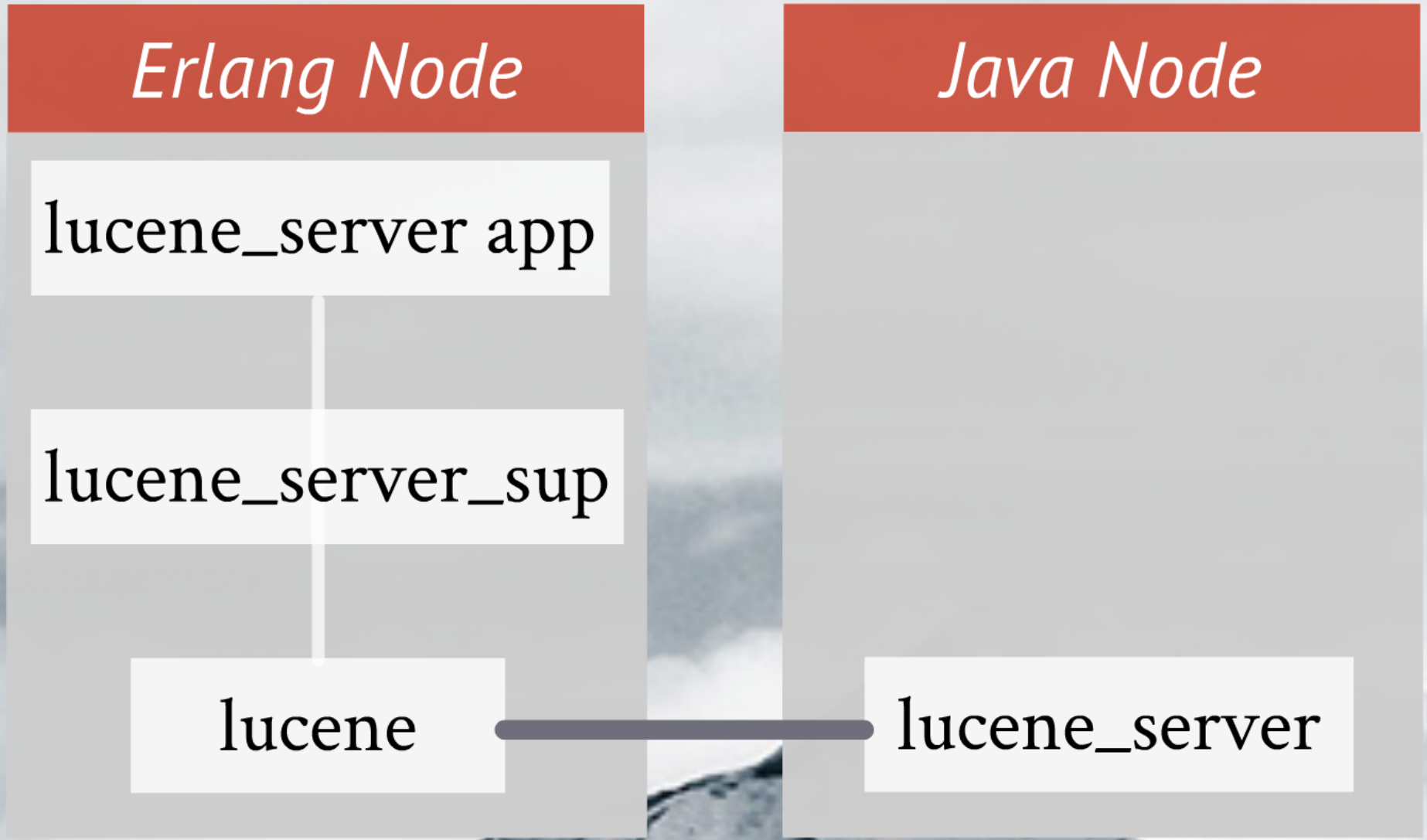
Rebar Compatibility

Make sure we have the right jar for jinterface in ./priv

```
{pre_hooks, [{compile, "mkdir -p bin"},
             {compile, "./copy-jinterface.sh"}]}.

{post_hooks,
  [{clean,
    "rm -rf bin priv/lucene_server.jar priv/OtpErlang.jar"},
   {compile,
    "javac -g -deprecation -sourcepath java_src
           -classpath ./bin:./priv/* -d bin
           java_src/*/*/*/*.java"},
   {compile, "jar cf priv/lucene-server.jar -C bin ."}]}.
```

Compile JAVA code and generate a jar file

Application Architecture



Lucene (the gen_server)

```
init([]) ->
...
  Port =
    erlang:open_port(
      {spawn_executable, Java},
      [{line,1000}, stderr_to_stdout,
       {args, JavaArgs ++
                ["-classpath", Classpath,
                 "com.tigertext.lucene.LuceneNode",
                 ThisNode, JavaNode, erlang:get_cookie(),
                 integer_to_list(AllowedThreads)]}]),
  wait_for_ready(
    #state{java_port = Port, java_node = JavaNode})
end.
...
wait_for_ready(State = #state{java_port = Port}) ->
  receive
    {Port, {data, {eol, "READY"}}}
      true = link(process()),
      true = erlang:monitor_node(State#state.java_node, true),
      {ok, State};
    Info ->
      ...
  end.
```

opens the Java process as a port

waits until it's ready

links to the lucene_server process in the Java node

monitors the Java node

LuceneNode

```
public static void main(String[] args) {
  String peerName = args.length >= 1 ? args[0]
      : "lucene_server@localhost";
  String nodeName = args.length >= 2 ? args[1]
      : "lucene_server_java@localhost";

  try {
    NODE = args.length >= 3 ? new OtpNode(nodeName, args[2])
        : new OtpNode(nodeName);
    PEER = peerName;

    final OtpGenServer server = new LuceneServer(NODE);

    server.start();

    System.out.println("READY");

  } catch (IOException e1) {
    jlog.severe("Couldn't create node: " + e1);
    e1.printStackTrace();
    System.exit(1);
  }
}
```

Starts a new OtpNode

Starts a new LuceneServer in it

Signals its ready state

A gen_server in Java

Introduction

We had a process on the Java node that received messages and, for some of them, it *returned* a value.

That looked a lot like a **gen_server**.

We also wanted all the functionality gen_server provides for its clients

code in gen.erl

```
do_call(Process, Label, Request, Timeout) ->
    try erlang:monitor(process, Process) of
        Mref ->
...
    catch
        error:_ ->
            %% Node (C/Java?) is not supporting the monitor.
            %% The other possible case -- this node is not
            %% distributed -- should have been handled earlier.
            %% Do the best possible with monitor_node/2.
            %% This code may hang indefinitely if the Process
            %% does not exist. It is only used for featureweak
            %% remote nodes.
            Node = get_node(Process),
            monitor_node(Node, true),
...
    end.
```

So, there are gen servers in Java, right?

OtpGenServer

abstract class == behavior

```
public abstract class OtpGenServer extends OtpSysProcess {
  protected OtpGenServer(OtpNode host) {
    super(host);
  }

  protected OtpGenServer(OtpNode host, String name) {
    super(host, name);
  }
...
  protected abstract OtpErlangObject handleCall(
      OtpErlangObject cmd, OtpErlangTuple from)
      throws OtpStopException, OtpContinueException,
      OtpErlangException;

  protected abstract void handleCast(OtpErlangObject cmd)
      throws OtpStopException, OtpErlangException;

  protected abstract void handleInfo(OtpErlangObject cmd)
      throws OtpStopException;

  protected abstract void terminate(OtpErlangException oee);

}
```

constructors == start_link / init

abstract methods == behaviour callbacks

LuceneServer

```
protected OtpErlangObject handleCall(OtpErlangObject cmd,
    OtpErlangTuple from)
    throws OtpStopException, OtpContinueException,
    OtpErlangException {

  OtpErlangTuple cmdTuple = (OtpErlangTuple) cmd;
  OtpErlangAtom cmdName = (OtpErlangAtom) cmdTuple.elementAt(0);

  if (cmdName.atomValue().equals("pid")) {
    return super.getSelf();
  } else if (cmdName.atomValue().equals("match")) {
    String queryString =
      ((OtpErlangString) cmdTuple.elementAt(1)).stringValue();
    int pageSize =
      ((OtpErlangLong) cmdTuple.elementAt(2)).intValue();

    runMatch(queryString, pageSize, from);

    throw new OtpContinueException();

  } else
...
}
```

{pid}

{reply, self(), State}

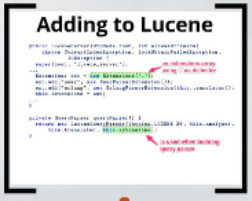
{match, QueryString, PageSize}

{noreply, State}

JInterface-stdlib

- Open-source library
- OTP/rebar compatible
- Includes:
  - **OtpSysProcess:** like gen module
  - **OtpGenServer:** like gen_server module

Extensions



.near

- **NearParserExtension** is built around lucene-spatial library
- Indexes fields using **n-tiers**
- We added a parser for **#geo{}** records

.erlang

Extensions

.near

Special value type:
#geo{lat :: float(), lon :: float()}

parameters: lat,lon,dist (in miles)

```
9> lucene:add(
    [[{index, I},{location, lucene_utils:geo(I/2,I+1.5)}]
     || I <- lists:seq(-10,10)]).
ok

10> lucene:match("location.near:0.0,1.0,100", 3).
{[[{index,0},
   {location,{geo,-8.381903171539307e-8,1.5000000782310963}},
   {''score'',-17.292743682861328}],
  [{index,-1},
   {location,{geo,-0.5000000540167093,0.500000137835741}},
   {''score'',-24.45547866821289}]],
 [{total_hits,2},
  {first_hit,1},
  {query_time,1984},
  {search_time,1021}]}
```

.erlang

an Erlang function

arguments (last one is always a list of values for the chosen field)

```
11> lucene:match(
      "index.erlang:\"lists:map:[fun(X) -> X*1.0 end]\"", 3).
{[[{index,10},
   {location,{geo,4.999999953433871,11.500000152736902}},
   {''score'',5.0}],
  [{index,9},
   {location,{geo,4.499999983236194,10.49999987706542}},
   {''score'',4.499999523162842}],
  [{index,8},
   {location,{geo,4.000000013038516,9.499999936670065}},
   {''score'',3.9999997615814213}]],
 [{total_hits,21},
  {first_hit,1},
  {query_time,2256},
  {search_time,1240},
  {next_page,<<172,237,0,5,115,114,0,36,99,111,109,46,
               116,105,103,101,114,116,101,120,...>>}]}
```

.near

Special value type:
#geo{lat :: float(), lon :: float()}

```
9> lucene:add(
     [[{index, I},{location, lucene_utils:geo(I/2,I+1.5)}]
      || I <- lists:seq(-10,10)]).
ok
```

parameters: lat,lon,dist (in miles)

```
10> lucene:match("location.near:0.0,1.0,100", 3).
{[[{index,0},
   {location,{geo,-8.381903171539307e-8,1.5000000782310963}},
   {''score',-17.2927436828613281}],
  [{index,-1},
   {location,{geo,-0.5000000540167093,0.5000001378357410}},
   {''score',-24.4554786682128906}]],
 [{total_hits,2},
  {first_hit,1},
  {query_time,1984},
  {search_time,1021}]}
```

.erlang

arguments (last one is always a list of values for the chosen field)

an Erlang function

```
11> lucene:match(
      "index.erlang:\"lists:map:[fun(X) -> X*1.0 end]\"", 3).
{[[{index,10},
   {location,{geo,4.999999953433871,11.500000152736902}},
   {'score',5.0}],
  [{index,9},
   {location,{geo,4.499999983236194,10.499999987706542}},
   {'score',4.499999523162842}],
  [{index,8},
   {location,{geo,4.000000013038516,9.499999936670065}},
   {'score',3.999999761581421}]],
 [{total_hits,21},
  {first_hit,1},
  {query_time,2256},
  {search_time,1240},
  {next_page,<<172,237,0,5,115,114,0,36,99,111,109,46,
               116,105,103,101,114,116,101,120,...>>}]}
```

Extensions

Adding to Lucene

```
public LuceneServer(OtpNode host, int allowedThreads)
    throws CorruptIndexException, LockObtainFailedException,
           IOException {
  super(host, "lucene_server");
...
  Extensions ext = new Extensions('.');
  ext.add("near", new NearParserExtension());
  ext.add("erlang", new ErlangParserExtension(this.translator));
  this.extensions = ext;
...
}

private QueryParser queryParser() {
  return new LuceneQueryParser(Version.LUCENE_36, this.analyzer,
      this.translator, this.extensions);
}
```

an extensions array using '.' as delimiter

is used when building query parser

.near

- **NearParserExtension** is built around lucene-spatial library
- Indexes fields using **n-tiers**
- We added a parser for **#geo{}** records

.erlang

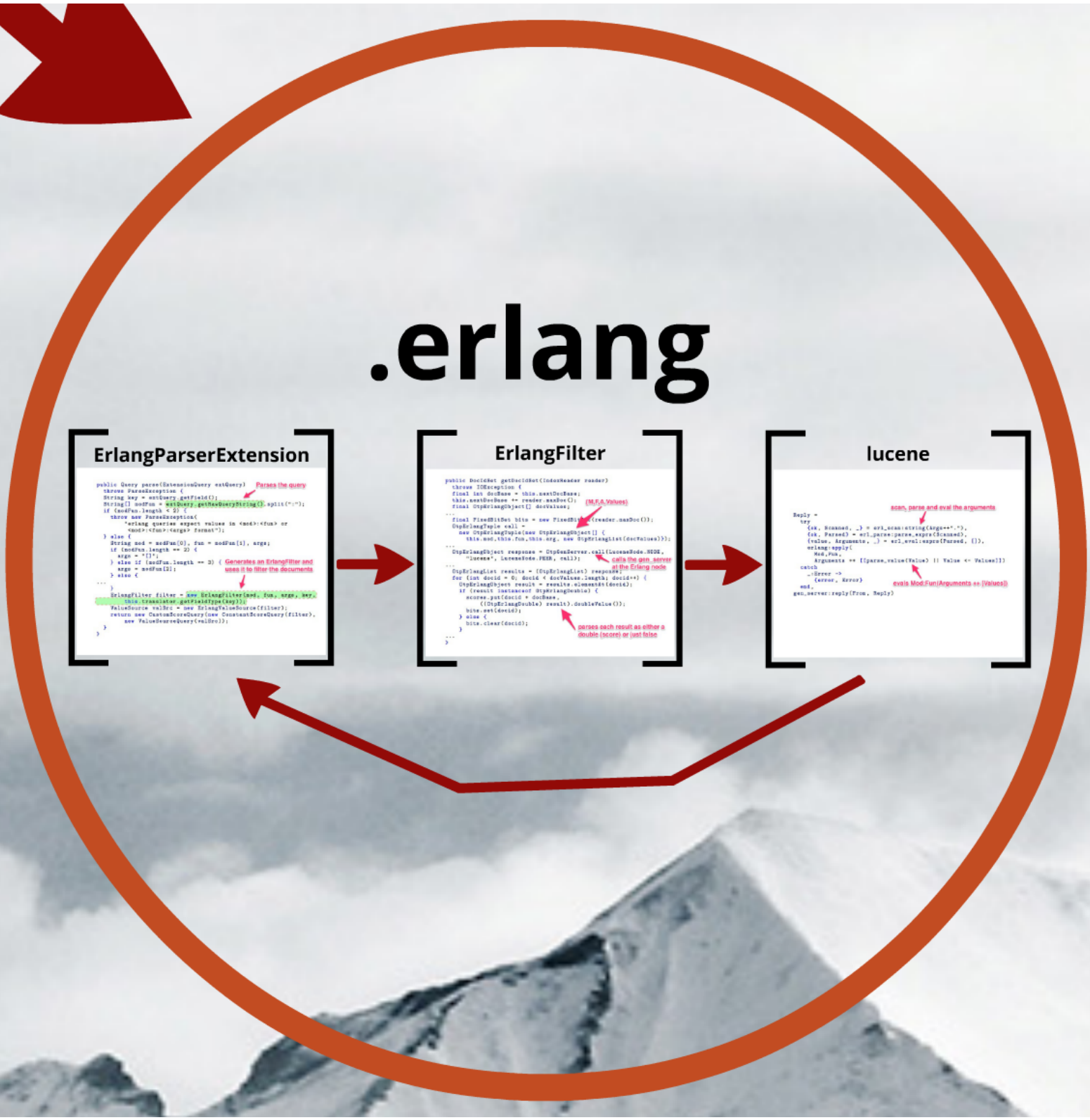
Adding to Lucene

```
public LuceneServer(OtpNode host, int allowedThreads)
    throws CorruptIndexException, LockObtainFailedException,
           IOException {
  super(host, "lucene_server");
...
  Extensions ext = new Extensions('.');
  ext.add("near", new NearParserExtension());
  ext.add("erlang", new ErlangParserExtension(this.translator));
  this.extensions = ext;
...
}

private QueryParser queryParser() {
  return new LuceneQueryParser(Version.LUCENE_36, this.analyzer,
      this.translator, this.extensions);
}
```

an extensions array using '.' as delimiter

is used when building query parser


.near
• NearParserExtension is built around lucene-spatial library
• Indexes fields using n-tiers
• We added a parser for #geo{} records


.erlang
ErlangParserExtension
Parses the query
Generates an ErlangFilter and uses it to filter the documents
ErlangFilter
(M,F,A,Values)
calls the gen_server at the Erlang node
parses each result as either a double (score) or just false
lucene
scan, parse and eval the arguments
evals Mod:Fun(Arguments ++ [Values])

ErlangParserExtension

```
public Query parse(ExtensionQuery extQuery)
  throws ParseException {
  String key = extQuery.getField();
  String[] modFun = extQuery.getRawQueryString().split(":");
  if (modFun.length < 2) {
    throw new ParseException(
        "erlang queries expect values in <mod>:<fun> or
         <mod>:<fun>:<args> format");
  } else {
    String mod = modFun[0], fun = modFun[1], args;
    if (modFun.length == 2) {
      args = "[]";
    } else if (modFun.length == 3) {
      args = modFun[2];
    } else {
...
    }
    ErlangFilter filter = new ErlangFilter(mod, fun, args, key,
        this.translator.getFieldType(key));
    ValueSource valSrc = new ErlangValueSource(filter);
    return new CustomScoreQuery(new ConstantScoreQuery(filter),
        new ValueSourceQuery(valSrc));
  }
}
```

Parses the query

Generates an ErlangFilter and uses it to filter the documents

ErlangFilter

```
public DocIdSet getDocIdSet(IndexReader reader)
  throws IOException {
  final int docBase = this.nextDocBase;
  this.nextDocBase += reader.maxDoc();
  final OtpErlangObject[] docValues;
...
  final FixedBitSet bits = new FixedBitSet(reader.maxDoc());
  OtpErlangTuple call =
    new OtpErlangTuple(new OtpErlangObject[] {
      this.mod,this.fun,this.arg, new OtpErlangList(docValues)});
...
  OtpErlangObject response = OtpGenServer.call(LuceneNode.NODE,
      "lucene", LuceneNode.PEER, call);
...
  OtpErlangList results = (OtpErlangList) response;
  for (int docid = 0; docid < docValues.length; docid++) {
    OtpErlangObject result = results.elementAt(docid);
    if (result instanceof OtpErlangDouble) {
      scores.put(docid + docBase,
          ((OtpErlangDouble) result).doubleValue());
      bits.set(docid);
    } else {
      bits.clear(docid);
    }
...
}
```

{M,F,A,Values}

calls the gen_server at the Erlang node

parses each result as either a double (score) or just false

lucene

scan, parse and eval the arguments

```
Reply =
  try
    {ok, Scanned, _} = erl_scan:string(Args++"."),
    {ok, Parsed} = erl_parse:parse_exprs(Scanned),
    {value, Arguments, _} = erl_eval:exprs(Parsed, []),
    erlang:apply(
      Mod,Fun,
      Arguments ++ [[parse_value(Value) || Value <- Values]])
  catch
    _:Error ->
      {error, Error}
  end,
gen_server:reply(From, Reply)
```

evals Mod:Fun(Arguments ++ [Values])

Lessons Learned

Supervising Java processes is not easy (but not because of what you think)

Communication between Erlang and Java nodes is costly (again, not because of the obvious reasons)

Not every string in Erlang is an OtpErlangString in Java.

Messages to and from Java nodes are encoded using Erlang's **External Term Format**

Supervising Java processes is not easy (but not because of what you think)

Not ev
Erl

Our Goal

- We want to *supervise* the process on the Java node.
- The process is registered as **lucene_server**

First Try

On an Erlang node, this just works

```
1> erlang:monitor(
        process, {lucene_server, 'other_node@host.local'}).
#Ref<0.0.0.210>
2> erlang:monitor(
        process,
        {lucene_server, 'lucene_server_java@host.local'}).
** exception error: bad argument
     in function  monitor/2
        called as monitor(
           process,
           {lucene_server,'lucene_server_java@host.local'})
     in call from erlang:dmonitor_p/2
```

...but not on a Java node

Let's Link!

link/1 needs pids, so we get the pid with rpc

```
1> Pid = rpc:call(
     'other_node@host.local', erlang, whereis, [lucene_server]).
<10086.75.0>
2> link(Pid).
true
3> Pid2 = rpc:call(
     'lucene_server_java@priscilla.local', erlang,
     whereis, [lucene_server], 5000).
{badrpc,timeout}
```

...but there is no rpc on Java nodes

The Final Version

```
1> Pid = gen_server:call(
         {lucene_server, 'lucene_server_java@priscilla.local'},
         {pid}).
<10087.1.0>
2> link(Pid).
true
```

since lucene_server is an OtpGenServer, we can ask it for its own pid with a gen_server call

Communication between Erlang and Java nodes is costly (again, not because of the obvious reasons)

tring in
an


The Scenario
• .erlang slowed down our queries a lot
• The more docs on the index, the slower queries
• More docs on the index, using .erlang, meant more and/or bigger messages between the nodes
• We discarded internal Lucene issues

The Benchmarker

Erlang Side

Sends a timestamped message to Java

Computes the total time it takes for it to come back

```
-spec tick(binary()) -> proplists:proplist().
tick(Data) ->
  TS1 = ts(),
  ?JAVA_SERVER ! {tick, self(), Data, TS1, TS1},
  receive
    {tick, _Pid, Data, TS1, TS2} ->
      TS3 = ts(),
      [ {'erl->java', TS2 - TS1}
      , {'java->erl', TS3 - TS2}
      , {'total',     TS3 - TS1}
      , {'size',      byte_size(term_to_binary(Data))}]
  after 30000 ->
    throw(timeout)
  end.

ts() ->
  {X, Y, Z} = os:timestamp(),
  X * 1000000000 + Y * 1000 + trunc(Z / 1000).
```

Java Side

Echoes whatever it gets plus a timestamp

```
/** This runs inside a never-ending loop */
protected static boolean processMsg(
  OtpErlangObject msg, OtpMbox mbox)
    throws OtpErlangDecodeException {
  if (msg instanceof OtpErlangTuple) {
    OtpErlangTuple cmd = (OtpErlangTuple) msg;
    OtpErlangAtom cmdName = (OtpErlangAtom) cmd.elementAt(0);
    OtpErlangPid caller = (OtpErlangPid) cmd.elementAt(1);
    if (cmdName.atomValue().equals("tick")) {
      OtpErlangObject result =
        new OtpErlangTuple(new OtpErlangObject[] { cmdName,
          mbox.self(), cmd.elementAt(2), cmd.elementAt(3),
          new OtpErlangLong(System.currentTimeMillis()) });
      mbox.send(caller, result);
    }
    return true;
  }
  throw new OtpErlangDecodeException("Bad message");
}
```

Erlang Side

Sends a timestamped message to Java

```
-spec tick(binary()) -> proplists:proplist().
tick(Data) ->
  TS1 = ts(),
  ?JAVA_SERVER ! {tick, self(), Data, TS1, TS1},
  receive
    {tick, _Pid, Data, TS1, TS2} ->
      TS3 = ts(),
      [ {'erl->java', TS2 - TS1}
      , {'java->erl', TS3 - TS2}
      , {'total',     TS3 - TS1}
      , {'size',      byte_size(term_to_binary(Data))}]
  after 30000 ->
    throw(timeout)
  end.

ts() ->
  {X, Y, Z} = os:timestamp(),
  X * 1000000000 + Y * 1000 + trunc(Z / 1000).
```

Computes the total time it takes for it to come back

Java Side

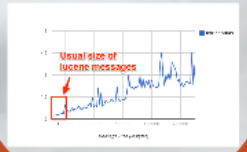
```
/** This runs inside a never-ending loop */
protected static boolean processMsg(
  OtpErlangObject msg, OtpMbox mbox)
    throws OtpErlangDecodeException {
  if (msg instanceof OtpErlangTuple) {
    OtpErlangTuple cmd = (OtpErlangTuple) msg;
    OtpErlangAtom cmdName = (OtpErlangAtom) cmd.elementAt(0);
    OtpErlangPid caller = (OtpErlangPid) cmd.elementAt(1);
    if (cmdName.atomValue().equals("tick")) {
      OtpErlangObject result =
        new OtpErlangTuple(new OtpErlangObject[] { cmdName,
          mbox.self(), cmd.elementAt(2), cmd.elementAt(3),
          new OtpErlangLong(System.currentTimeMillis()) });
      mbox.send(caller, result);
    }
    return true;
  }
  throw new OtpErlangDecodeException("Bad message");
}
```

Echoes whatever it gets plus a timestamp

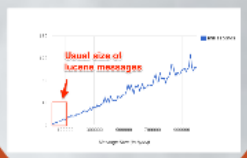
The Benchmarks

of Messages

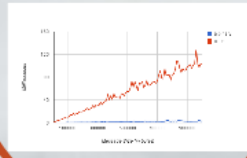
Size of Messages



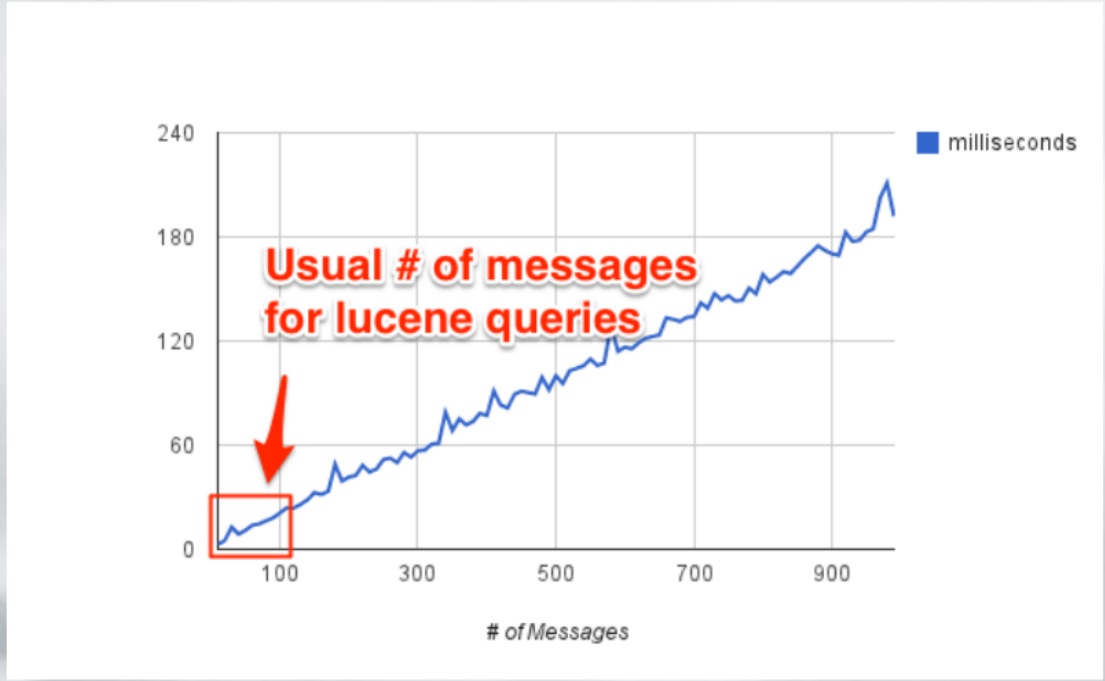
Length of Messages

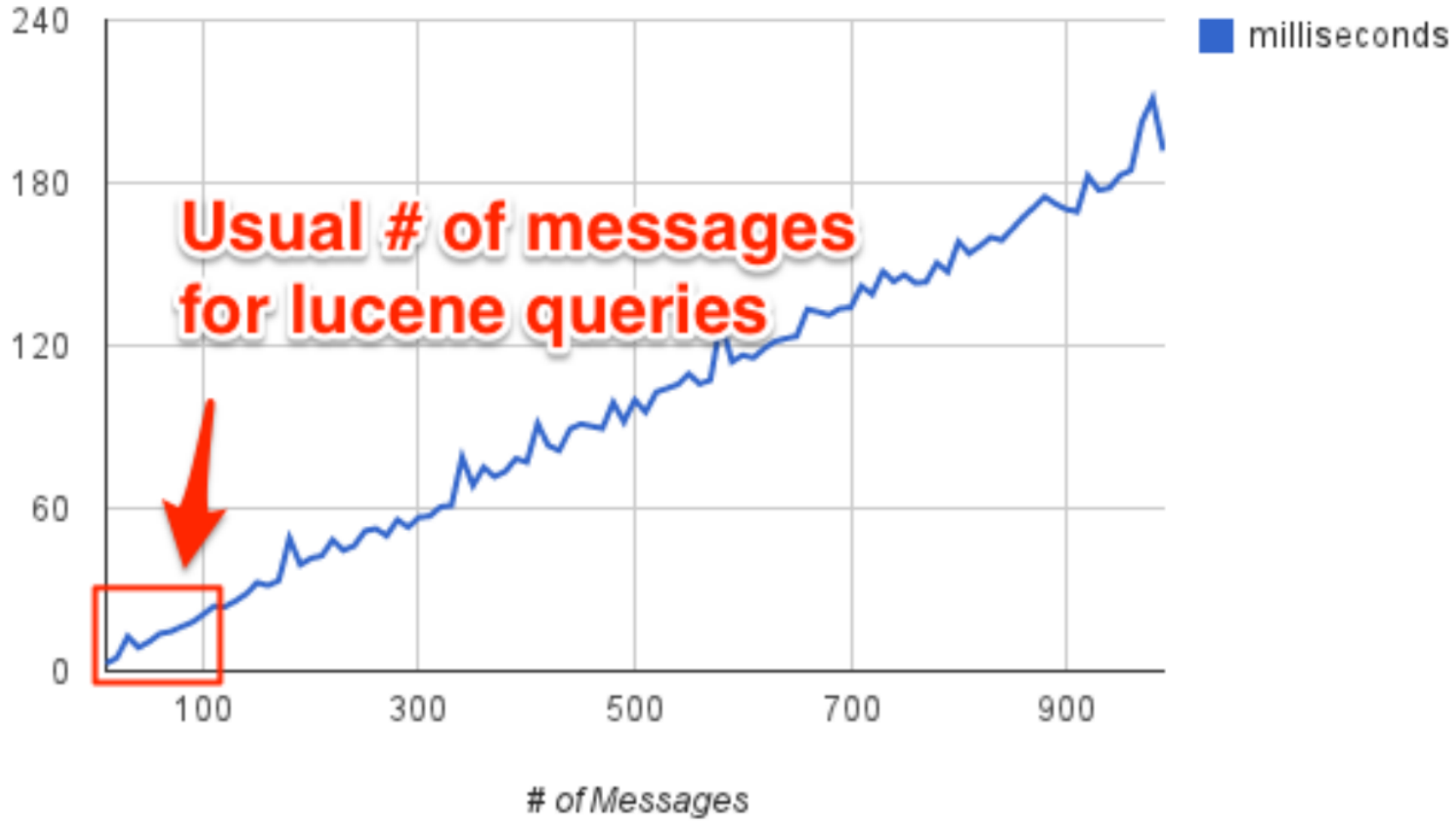


Side by Side

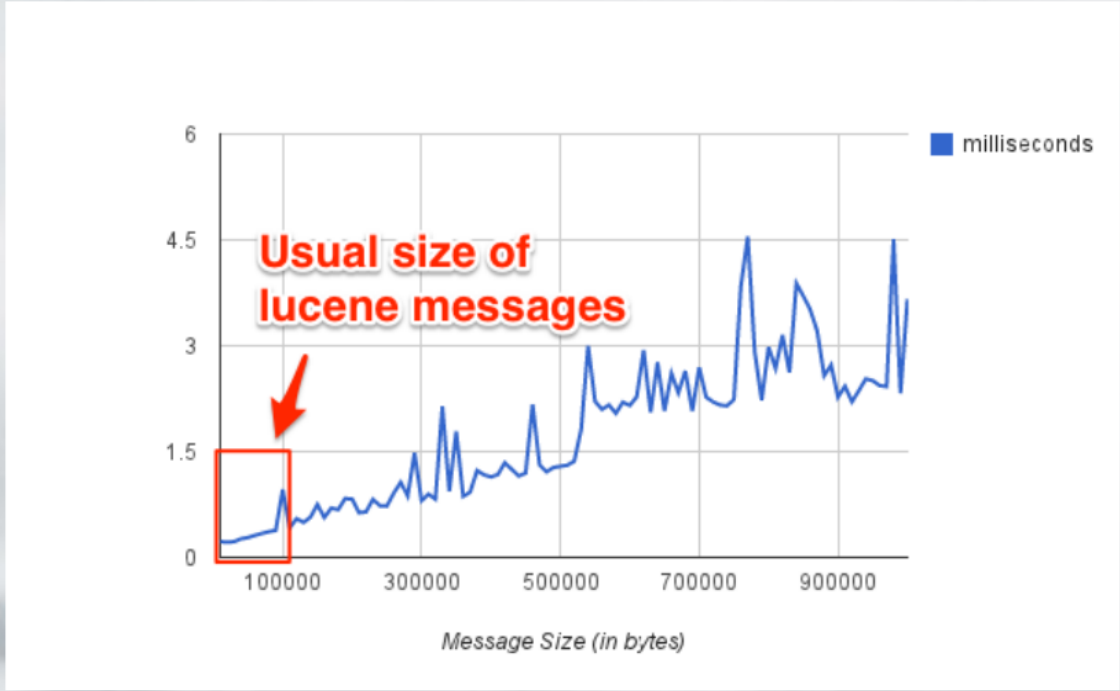


of Messages


240
180
120
60
0
100
300
500
700
900
milliseconds
Usual # of messages
for lucene queries
# of Messages

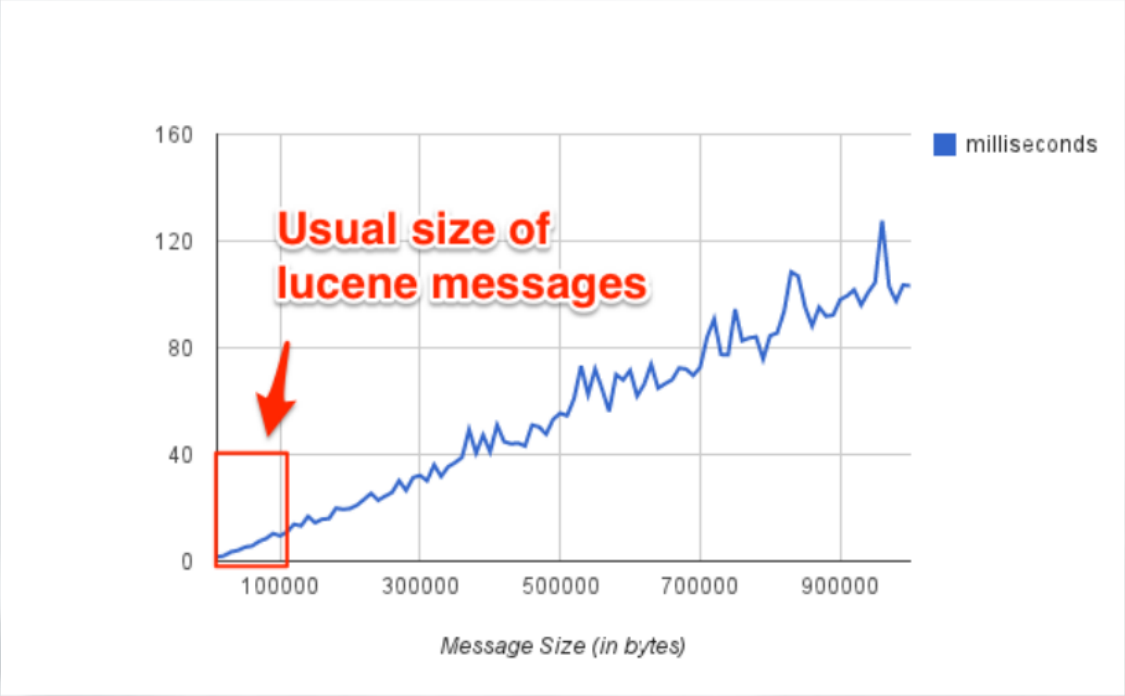

240
180
120
60
0
Usual # of messages
for lucene queries
100
300
500
700
900
# of Messages
milliseconds

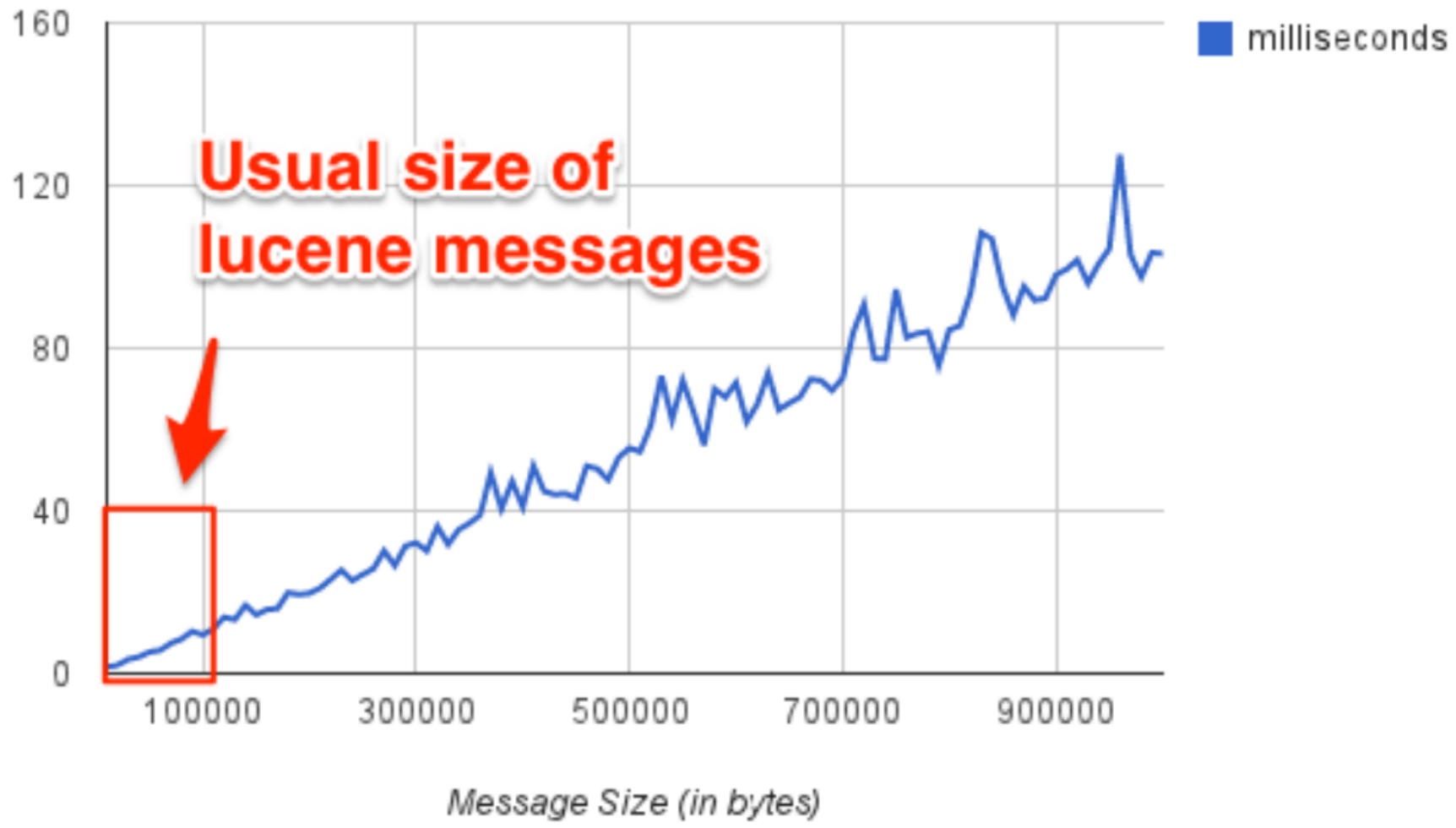
Size of Messages



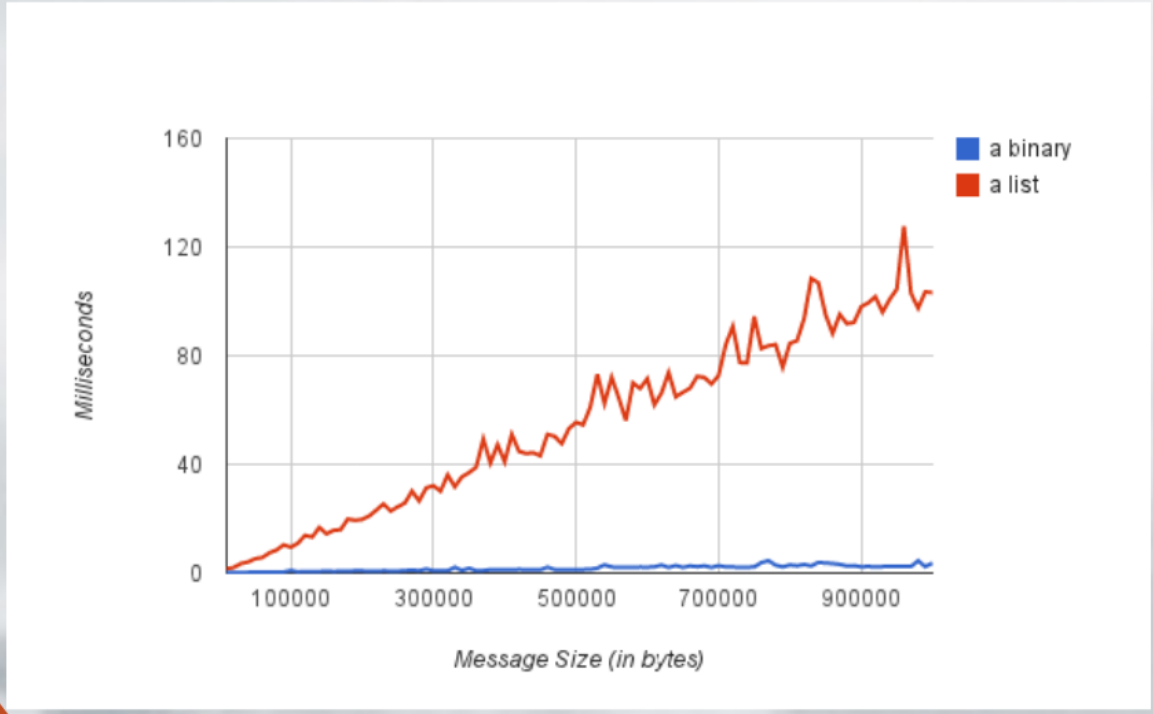

6
4.5
3
1.5
0
Usual size of
lucene messages
milliseconds
100000
300000
500000
700000
900000
Message Size (in bytes)

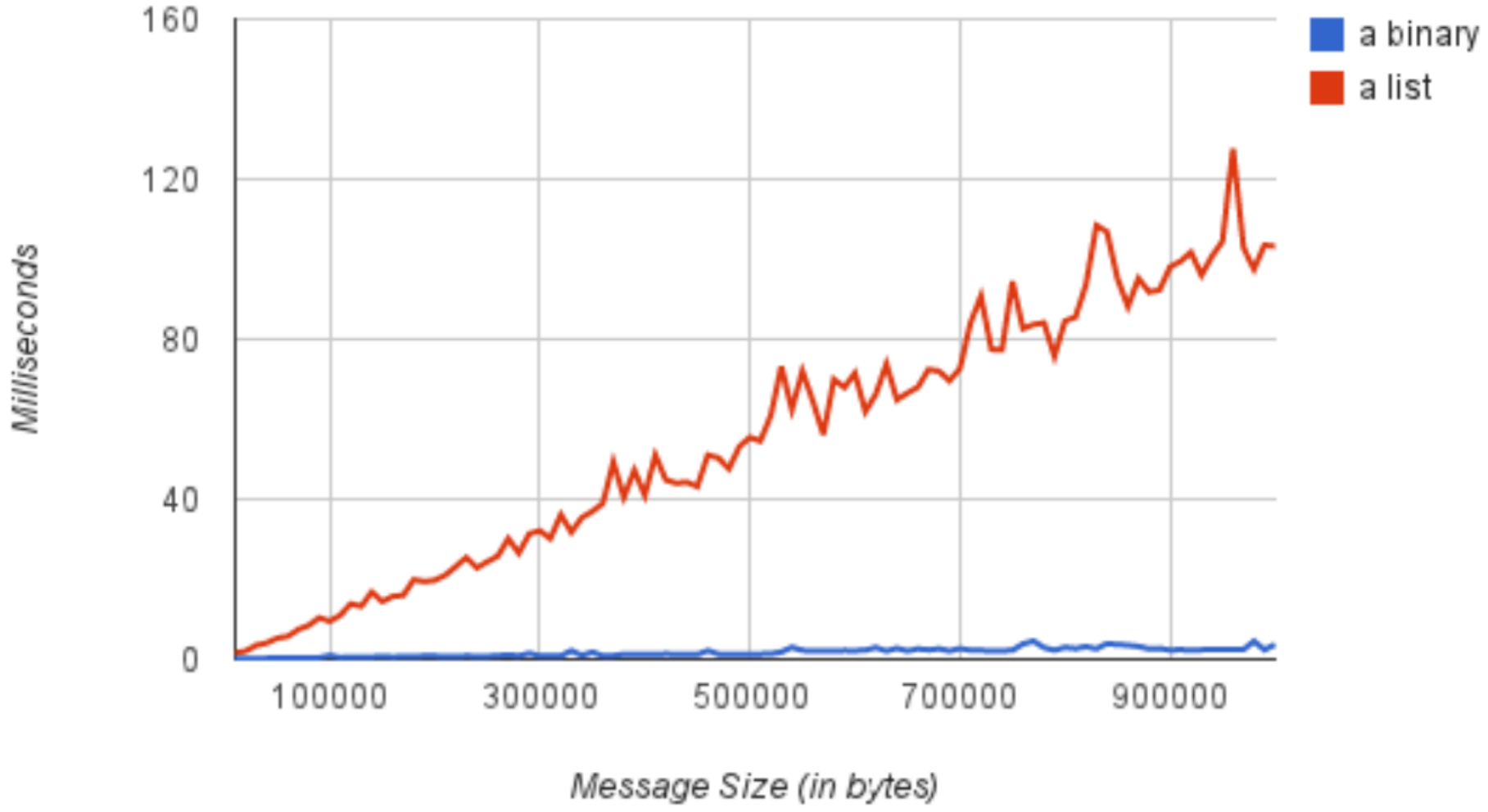
Length of Messages




160
120
80
40
0
Usual size of
lucene messages
milliseconds
100000
300000
500000
700000
900000
Message Size (in bytes)

Side by Side


160
120
80
40
0
Milliseconds
100000
300000
500000
700000
900000
Message Size (in bytes)
a binary
a list


160
120
80
40
0
Milliseconds
100000
300000
500000
700000
900000
Message Size (in bytes)
a binary
a list


Lesson
• It doesn't matter **how many** messages you exchange
• Message **size** doesn't matter
• The **number of objects** in the message **does**

Not every string in Erlang is an OtpErlangString in Java.

Messages to and from Java nodes are encoded using Erlang's **External Term Format**

The Docs

9.14 STRING_EXT

1	2	Len
107	Length	Characters

Table 9.20:

String does NOT have a corresponding Erlang representation, but is an optimization for sending lists of bytes (integer in the range 0-255) more efficiently over the distribution. Since the `Length` field is an unsigned 2 byte integer (big endian), implementations must make sure that lists longer than 65535 elements are encoded as **`LIST_EXT`**.

The Rules

Erlang	Length	Java
`""`	0	`OtpErlangList`
`"a string"`	8	`OtpErlangString`
`"a long string ..."`	> 65535	`OtpErlangList`

The Docs

9.14 STRING_EXT

1	2	Len
107	Length	Characters

Table 9.20:

String does NOT have a corresponding Erlang representation, but is an optimization for sending lists of bytes (integer in the range 0-255) more efficiently over the distribution. Since the `Length` field is an unsigned 2 byte integer (big endian), implementations must make sure that lists longer than 65535 elements are encoded as **`LIST_EXT`**.

The Rules

Erlang	Length	Java
""	0	OtpErlangList
"a string"	8	OtpErlangString
"a long string ..."	> 65535	OtpErlangList


The Future
• More extensions!
• Security improvements
• Other backends? (Currently it's just in-memory)
• More users! That's YOU!!

Resources

- github.com/tigertext/lucene_server
- github.com/inaka/jinterface_stdlib
- lucene.apache.org/core
- about.me/elbrujohalcon
- github.com/inaka
- inaka.net
- inaka.net/blog


THANK YOU VERY MUCH!!