What's new in Infinispan 7

Martin Genčúr

Tomáš Sýkora

JBug, Brno, April 8th 2015

Warm up

A few words about Infinispan

Deep dive

Infinispan Query API + Demo

Vision

Infinispan Management Console + Demo

+ bonus :)

What's Infinispan?

- Open-source datagrid patform
- In-Memory, Schema-less, NoSQL key-value data store
- Distributed cache (offers massive heap)
- Scalable (goal: hundreds of nodes) + Elastic
- Highly available, resilient to node failures
- Concurrent (reads and writes at the same time)
- Transactional
- Queryable (also document store)

What's Infinispan?

- Open-source datagrid patform
- **In-Memory, Schema-less, NoSQL key-value data store**
- Distributed cache (offers massive heap)
- Scalable (goal: hundreds of nodes) + Elastic
- Higly available, resilient to node failures
- Concurrent (reads and writes at the same time)
- Transactional
- Queryable (also document store)

What's Infinispan?

- Open-source datagrid patform
- In-Memory, Schema-less, NoSQL key-value data store
- **Distributed cache** (offers massive heap)
- Scalable (goal: hundreds of nodes) + Elastic
- Higly available, resilient to node failures
- Concurrent (reads and writes at the same time)
- Transactional
- Queryable (also document store)

What's Infinispan?

- Open-source datagrid patform
- In-Memory, Schema-less, NoSQL key-value data store
- Distributed cache (offers massive heap)
- **Scalable** (goal: hundreds of nodes) + **Elastic**
- Higly available, resilient to node failures
- Concurrent (reads and writes at the same time)
- Transactional
- Queryable (also document store)

What's Infinispan?

- Open-source datagrid patform
- In-Memory, Schema-less, NoSQL key-value data store
- Distributed cache (offers massive heap)
- Scalable (goal: hundreds of nodes) + Elastic
- **Higly available, resilient to node failures**
- Concurrent (reads and writes at the same time)
- Transactional
- Queryable (also document store)

What's Infinispan?

- Open-source datagrid patform
- In-Memory, Schema-less, NoSQL key-value data store
- Distributed cache (offers massive heap)
- Scalable (goal: hundreds of nodes) + Elastic
- Higly available, resilient to node failures
- **Concurrent** (reads and writes at the same time)
- **Transactional**
- Queryable (also document store)

What's Infinispan?

- Open-source datagrid patform
- In-Memory, Schema-less, NoSQL key-value data store
- Distributed cache (offers massive heap)
- Scalable (goal: hundreds of nodes) + Elastic
- Higly available, resilient to node failures
- Concurrent (reads and writes at the same time)
- Transactional
- **Queryable** (also document store)

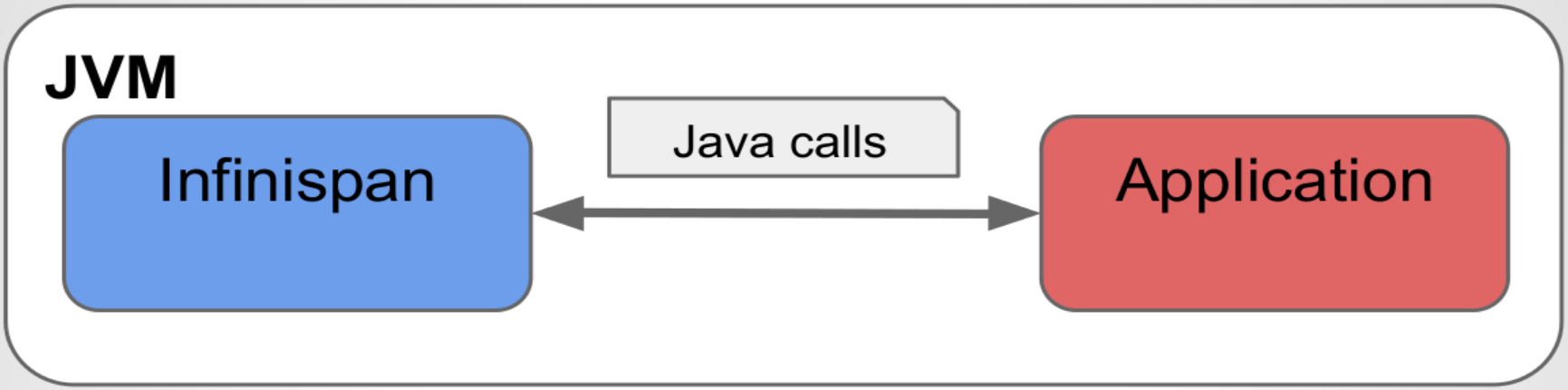
For Java users: it's a Map

Yes, just a Map – everything else is for free!

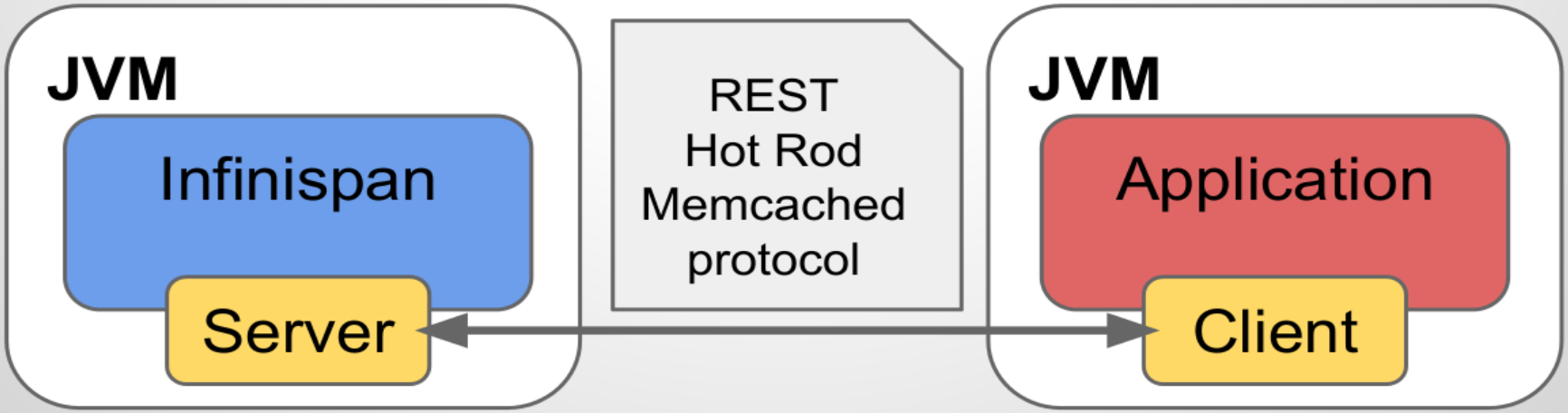
```
DefaultCacheManager cacheManager =
        new DefaultCacheManager("infinispan.xml");

Cache<String, Object> cache =
        cacheManager.getCache("namedCache");

cache.put("key", "value");

Object value = cache.get("key");
```


Interacting with Infinispan

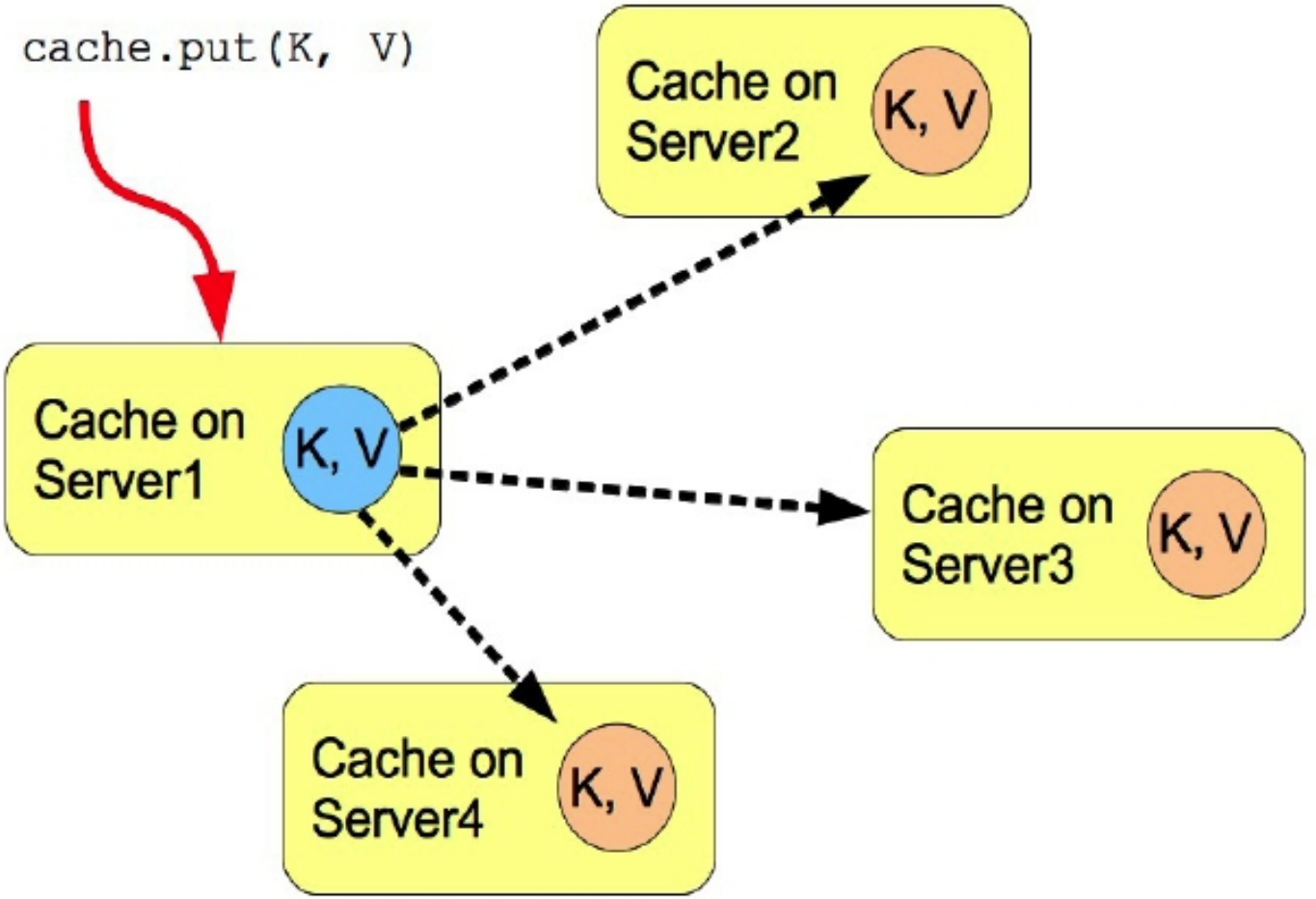
- InVM, Embedded, Library mode



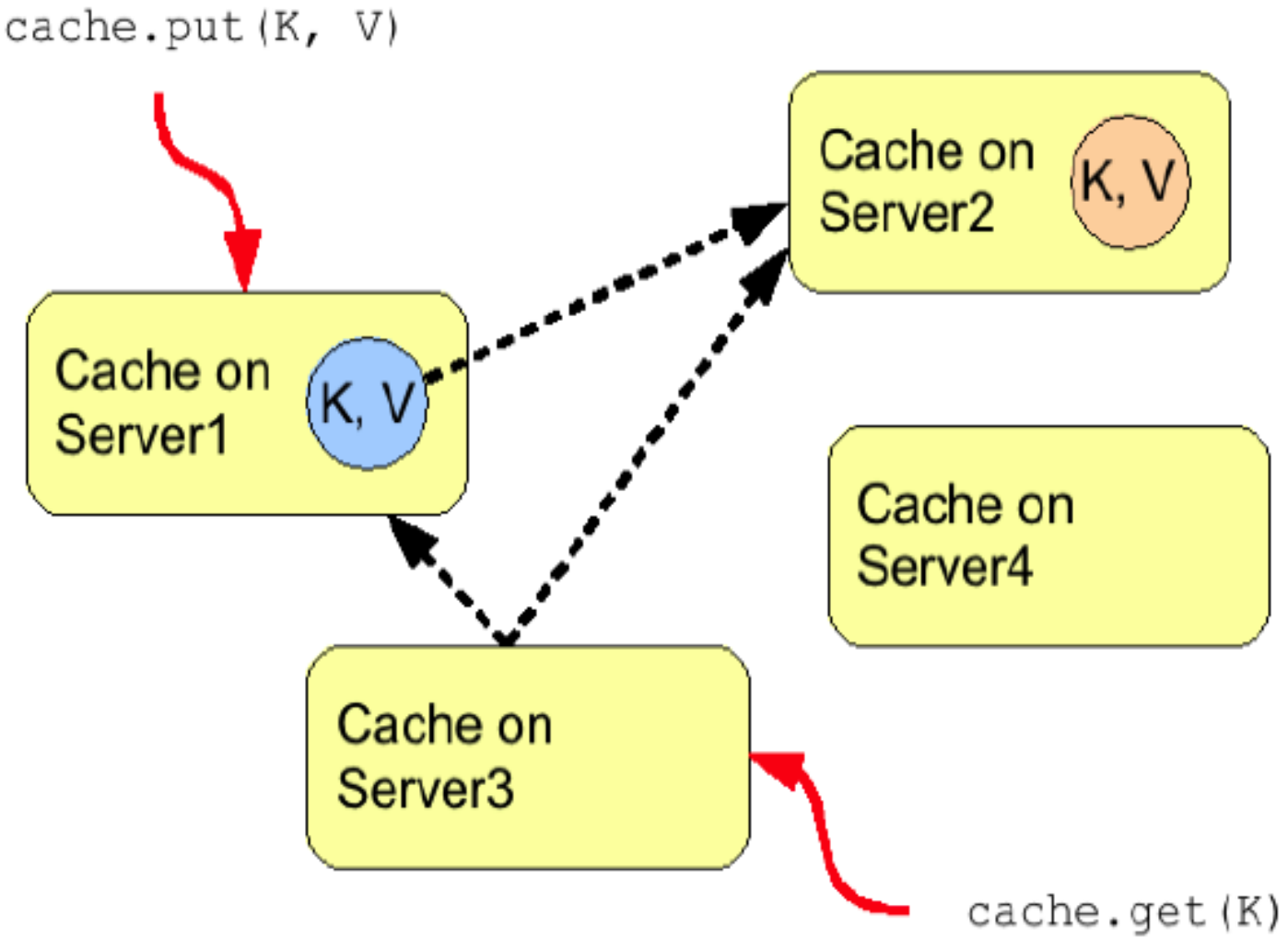
- Client-server mode



Replication mode



Distribution mode



Features - Querying

Map / Key-Value approach:

Key1 --> Value1
Key2 --> Value2 (dictionary like)

Querying mechanism allows you to query Infinispan over the values!

? “data where name='infinispan' and ...” --> **Value1, Value2**

Infinispan Query

Infinispan Query

- embedded querying existing for a long time
- POJOs annotated with Hibernate Search annotations (@Indexed, @Field)
- Lucene queries and indexing

Embedded Query

```
QueryBuilder queryBuilder = sm.buildQueryBuilderForClass(Car.class).get();
    return queryBuilder
                .bool()
                .should(queryBuilder.keyword().onFields("brand")
                     .matching(car.getBrand())
                     .createQuery())
                .must(queryBuilder.keyword().onFields("color")
                      .matching(car.getColor())
                      .createQuery())
                .createQuery();
```

- Need for client-server querying ...

Remote Query

- server written in Java, client need not
- binary protocol between client and server
- messages can't be just binary blobs
- language neutral querying based on internal DSL
- language neutral encoding

- Protobuf !

Remote Query – Protobuf Descriptors

```
message Person {
  required string name = 1; //float, double, bool, uint64, sint32, bytes
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phone = 4;
}
```


Protobuf API

- Java

```
byte raw_bytes[] = //allocate byte array
CodedOutputStream coded_output = new CodedOutputStream(raw_bytes);

int magic_number = 1234;
coded_output.writeLittleEndian32(magic_number);

byte text_bytes[] = "Hello world!".getBytes(charset);
coded_output.writeRawBytes(text_bytes);
```


Protobuf API

- C++

```
byte raw_bytes[] = //allocate byte array of size SIZE
ArrayOutputStream* array_stream = new ArrayOutputStream(raw_bytes, SIZE);
CodedOutputStream* coded_output = new CodedOutputStream(array_stream);

int magic_number = 1234;
coded_output->WriteLittleEndian32(magic_number);

char text[] = "Hello world!";
coded_output->WriteRaw(text, strlen(text));
```


Remote Query Syntax – Basic Structure

```
org.infinispan.query.dsl.Query query =
      qf.from(Book.class)
        .having("title").like("%engine%")
        .toBuilder().build();
```


Remote Query Syntax – Filtering Operators

- in [collection of values]
- [collection of values] contains [value]
- containsAll, containsAny
- isNull
- like
- eq, gt, gte, lt, lte
- between (e.g. .having("date").between(date1, date2))

Remote Query Syntax - Conditions

- and, or, not

```
.from(Book.class).not().having("title").like("%Infinispan%")
```

- nested condition example

```
.from(Book.class).having("author.name").eq("Manik")
```


Remote Query Syntax - Continued

- Projections
  - will return List of Object[]

```
.from(Book.class).setProjection("title", "publicationYear")
```

- Sorting

```
.from(Book.class).orderBy("publicationYear",
SortOrder.DESC)
```

- Pagination

```
.from(Book.class).setStartOffset(20).maxResults(10)
```


Infinispan 7 Enhancements

- Index-less Query
  - based on distributed iterators
  - all fields are searchable, not only the indexed ones
  - enabling index-less querying - just leave the remote cache un-indexed
  - mixed mode currently not available
  - faster for storing data, slower for querying

Infinispan 7 Enhancements - Continued

- Java annotations for easier indexing
  - @ProtoDoc, @ProtoMessage
  - @ProtoField
  - @ProtoEnum, @ProtoEnumValue
- No need to compile .proto to .protobin
- Protobuf descriptors registered via remote client over HotRod protocol
  - used to be a JMX operation

Remote Query Step by Step

1) Configure Cache for indexing on the server
2) Place annotations on domain objects
3) Configure and start remote HotRod client
4) Register .proto file with the client and server
5) Put data in the remote cache
6) Query your data !

Infinispan Management Console

Infinispan Management Console

- Web console
- Standalone management solution
- Very young project (4 months now)
- JavaScript and other Web technologies

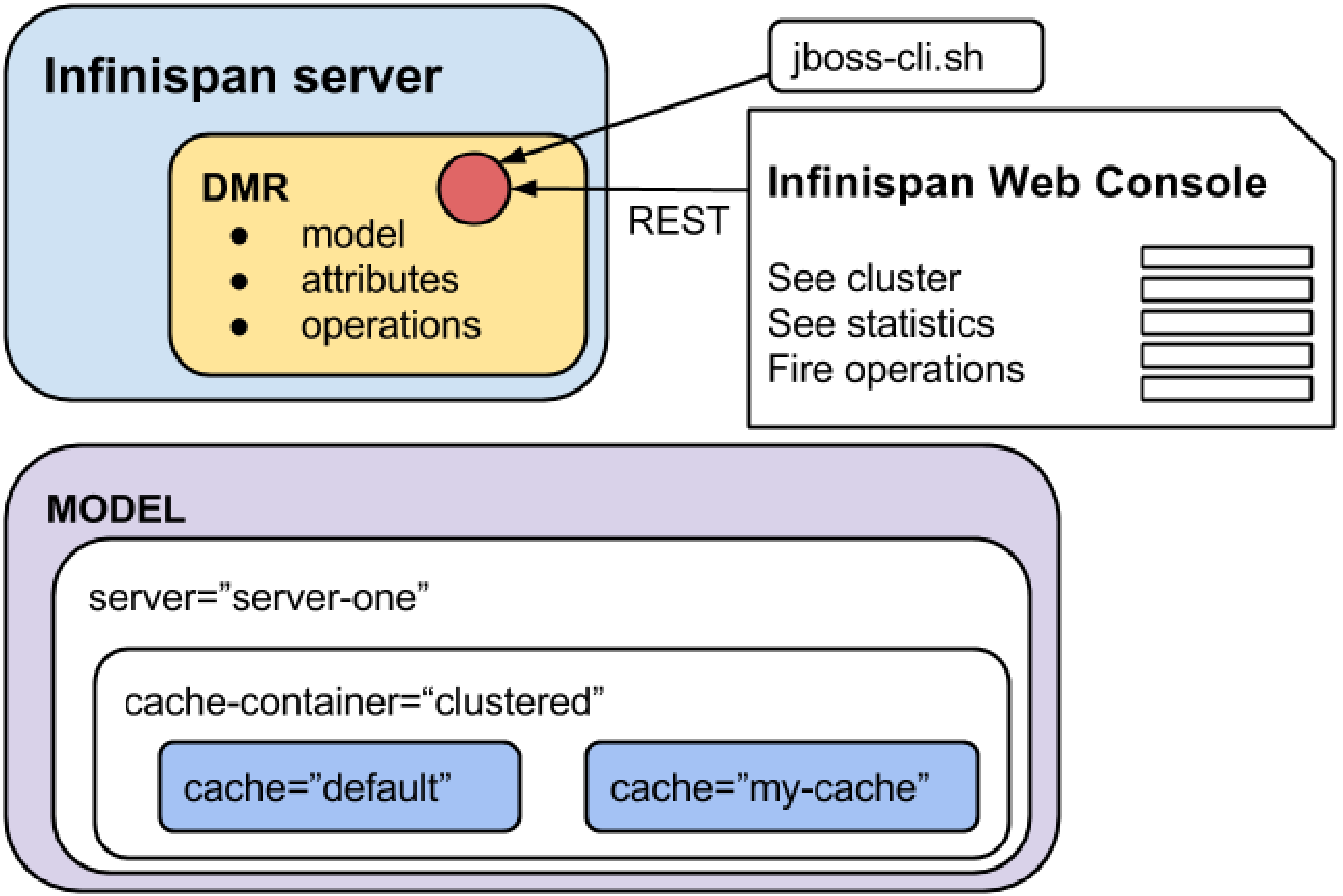
- Connected to the Infinispan internals via DMR

Infinispan Management Console

How do we obtain statistics / manage Infinispan?

- DMR (Dynamic Model Representation)
  - jboss-dmr library
  - Allowing to easily represent any
    - Resource
    - Attribute
    - Operation

- Infinispan internals expose statistics via JBoss DMR
- DMR is exposed to the Console via REST interface


Infinispan server
jboss-cli.sh
DMR
model
attributes
operations
REST
Infinispan Web Console
See cluster
See statistics
Fire operations
MODEL
server="server-one"
cache-container="clustered"
cache="default"
cache="my-cache"

DEMO

Bonus :)

JBoss R&D Laboratory at FI MUNI

- Dedicated room at faculty
- Students can participate in open-source projects
- Each project has tutor / leader
- Stipend provided by faculty (based on contribution)

- win-win situation
  - New contributors
  - Students gain experience

Graphical Network Communication Tracer

Our challenge

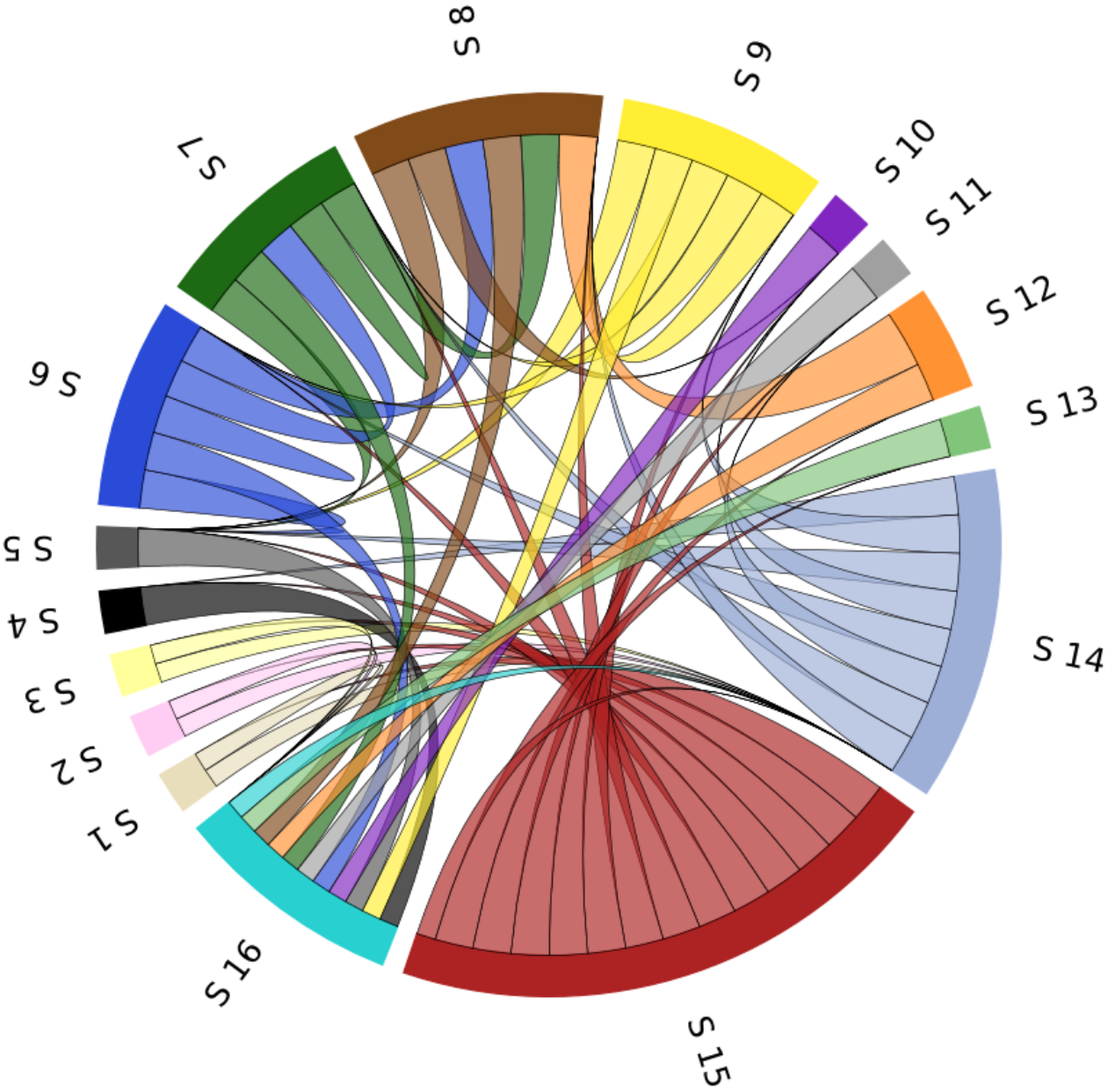
- Infinispan uses JGroups for internal communication
- Very hard to debug big trace logs
  - Terabytes of plain text

Our motivation

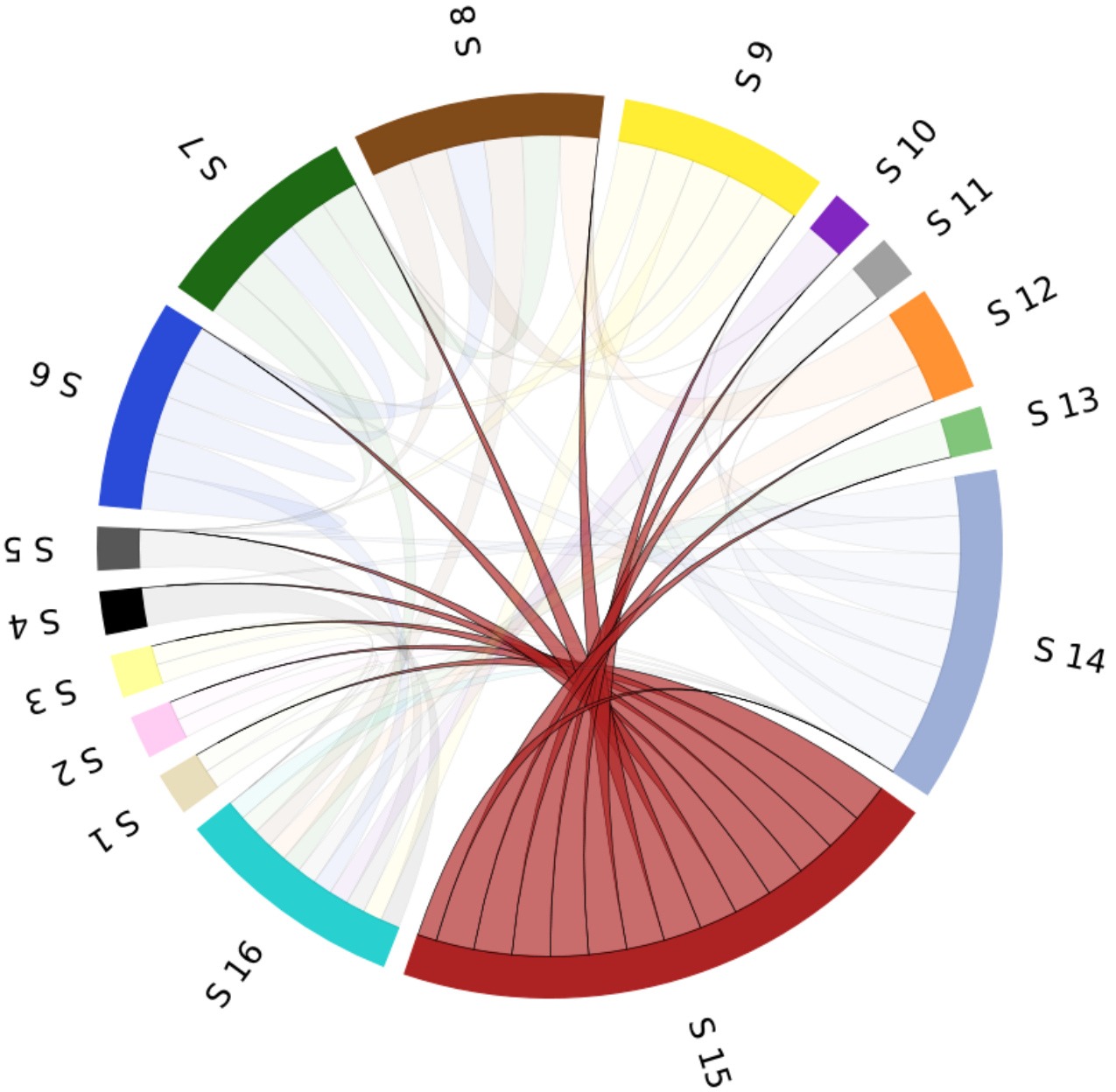
- Show message flows to users and developers
- Live! Now!

- Spot problems in a graphical interface
- Create and show traceable network history

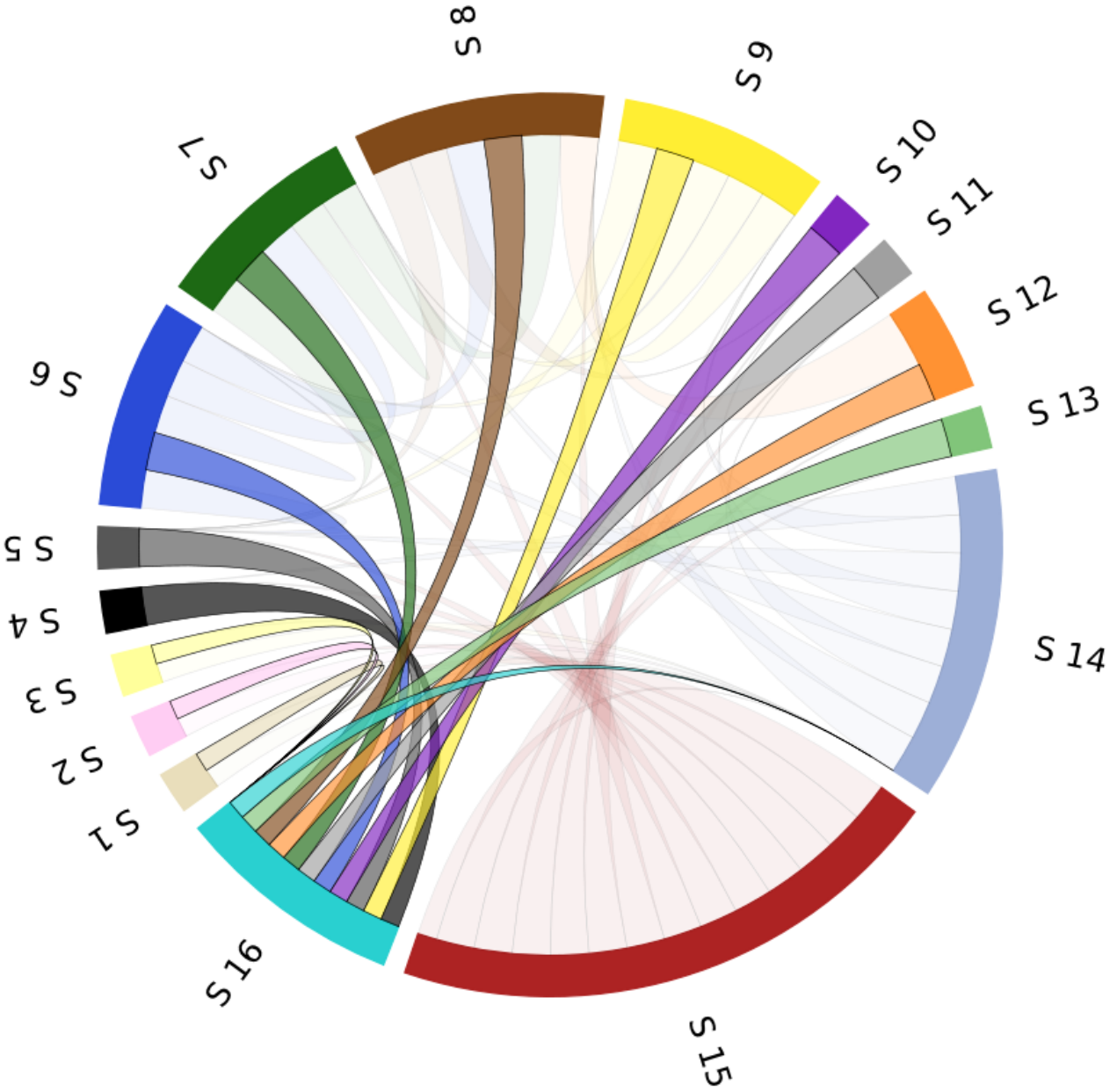
How it could look like?



How it could look like?



How it could look like?



What can you expect?

- JavaScript
- AngularJS
- Web technologies in general
- Infinispan and JGroups

- Designing UI
- Connecting to internal Infinispan processes
- Monitoring servers and network communication

Credits

Image sources:

http://commons.wikimedia.org/wiki/File:AngularJS_logo.svg
http://www.javatpoint.com/javascript-tutorial

https://developer.jboss.org/wiki/JBossRDLabAtFIMUNI

Q & A

mgencur@redhat.com
tsykora@redhat.com

THANK YOU!