springone 2GX CHICAGO 2011

Writing applications for Cloud Foundry using Spring and MongoDB

Thomas Risberg, VMware
<trisberg@vmware.com> - @trisberg

Jared Rosoff, 10gen
<jsr@10gen.com> - @forjared

springsource
A division of vmware

Outline

What we will cover:

- Introduction to Cloud Foundry
- Introduction to MongoDB
- Spring Data support for MongoDB
  - Using the MongoTemplate
  - Using the Mongo Repository support
- Why run MongoDB in the cloud?

What is Cloud Foundry?

- Cloud Foundry is a PaaS
  - The application platform will be delivered as a service in the cloud era
  - The industry calls this platform as a service (PaaS)
- PaaS makes it much easier to deploy, run and scale applications
- But PaaS solutions in the market have fatal flaws today
  - Limited in framework, application services and/or cloud support
- Cloud Foundry aim to fix that…

Characteristics of PaaS

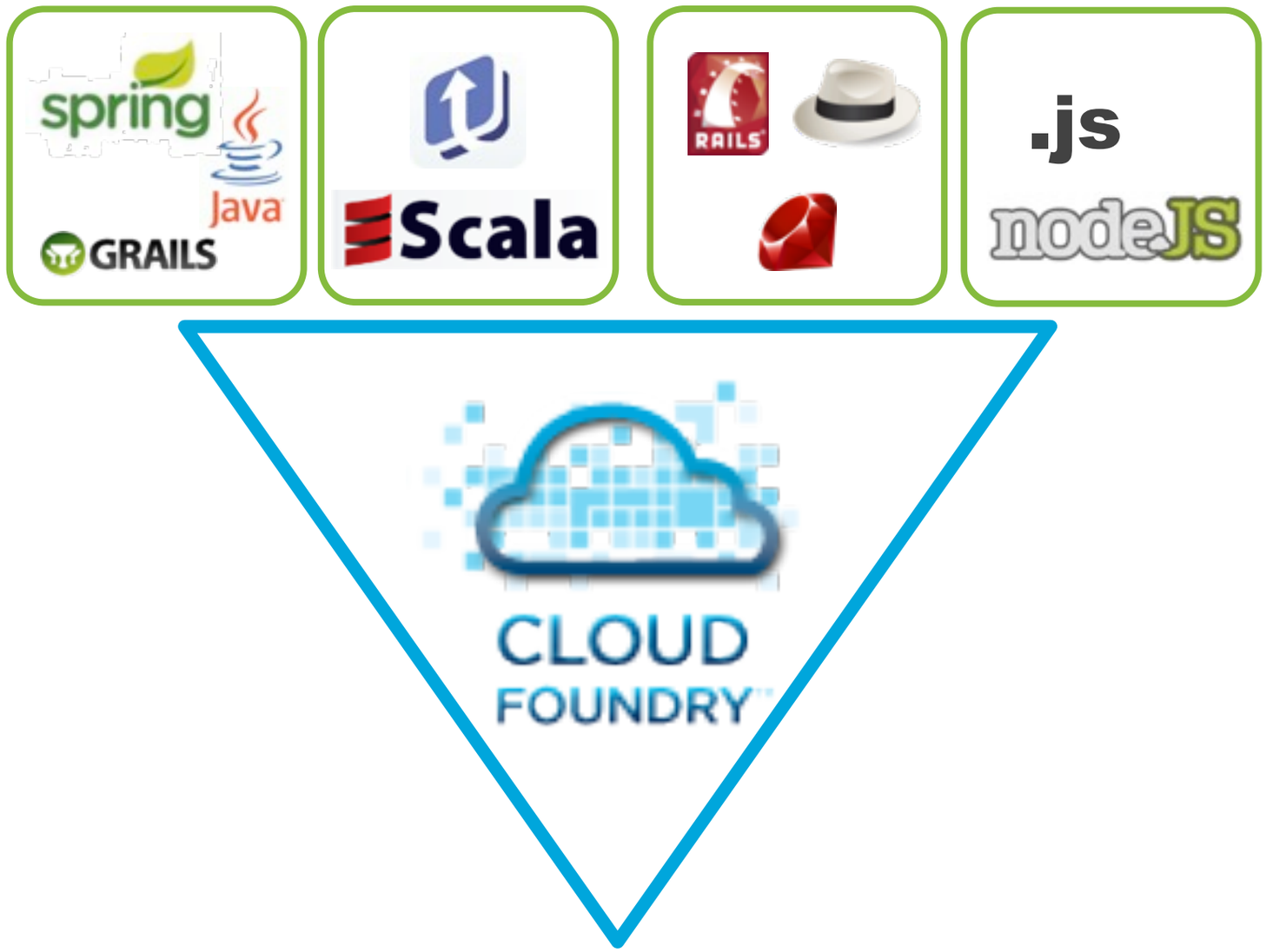
The application platform for the cloud era

- Integrated software stack
- Application execution engine
- Self-service application deployment
- Automated application infrastructure provisioning
- Curated, updated and operated as a service

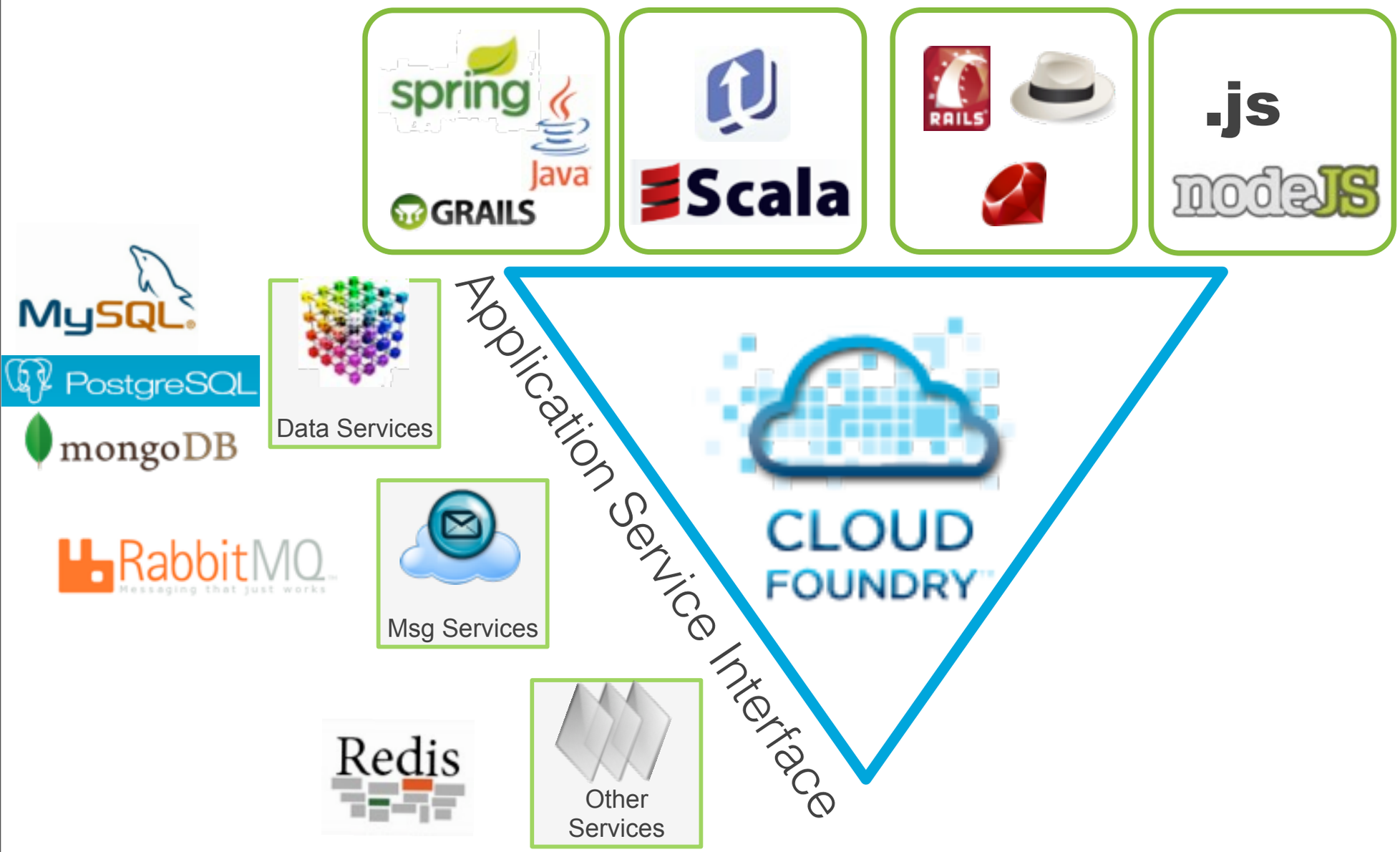
Cloud Foundry – The first open PaaS

- **Self-service application execution engine**
  - Build applications with latest high productivity frameworks
- **Automation engine for deployment and lifecycle management**
  - Deploy and cloud-scale applications in seconds
- **Open architecture**
  - Choice of clouds for deployment
  - Choice of industry-standard frameworks
  - Choice of application infrastructure services
  - Extensible architecture to “digest” future cloud innovation
  - Available as open source

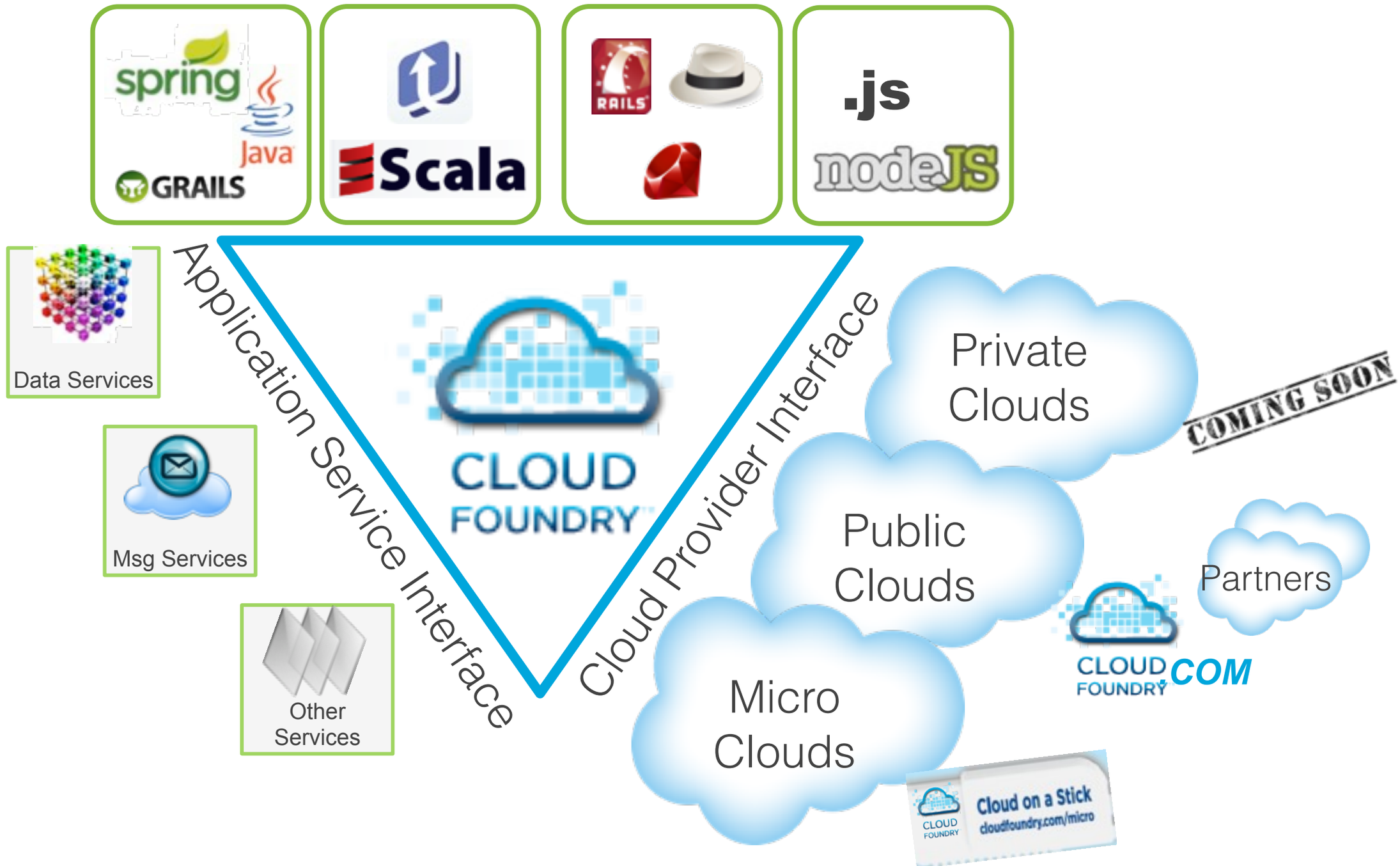
Choice of frameworks



Choice of application services



Choice of clouds




mongoDB

MongoDB

Application

High Performance

Document Oriented

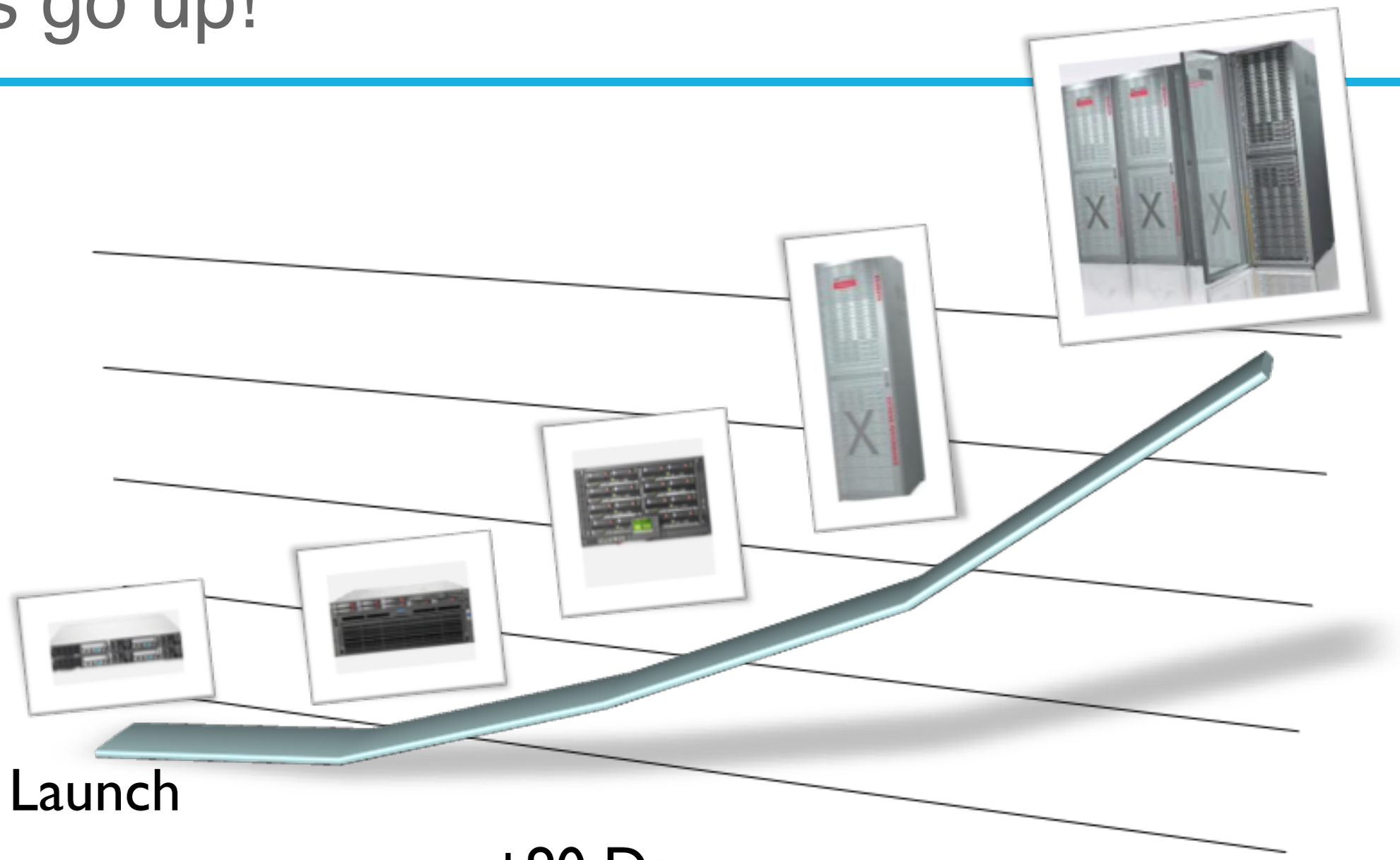
```
{ author:“steve”,
  date: new Date(),
  text: “About MongoDB...”,
  tags: [“tech”,“database”]}
```

Horizontally Scalable

Why MongoDB

- Double click serving 400,000 ads / second
- People writing their own datastores
- Caching is de rigueur
- Complex ORM frameworks
- Computer architecture trends
- Cloud computing

Costs go up!

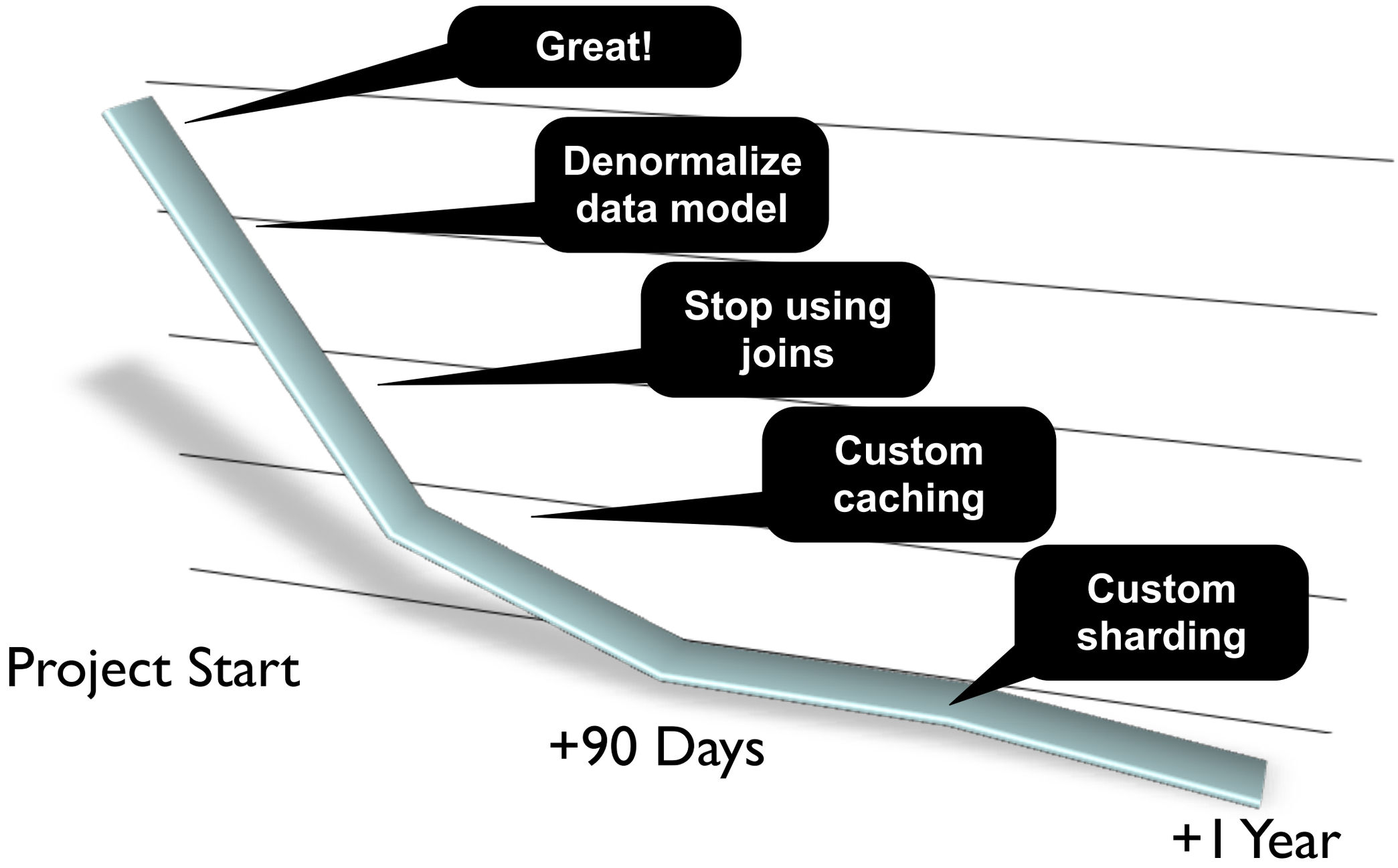


Launch

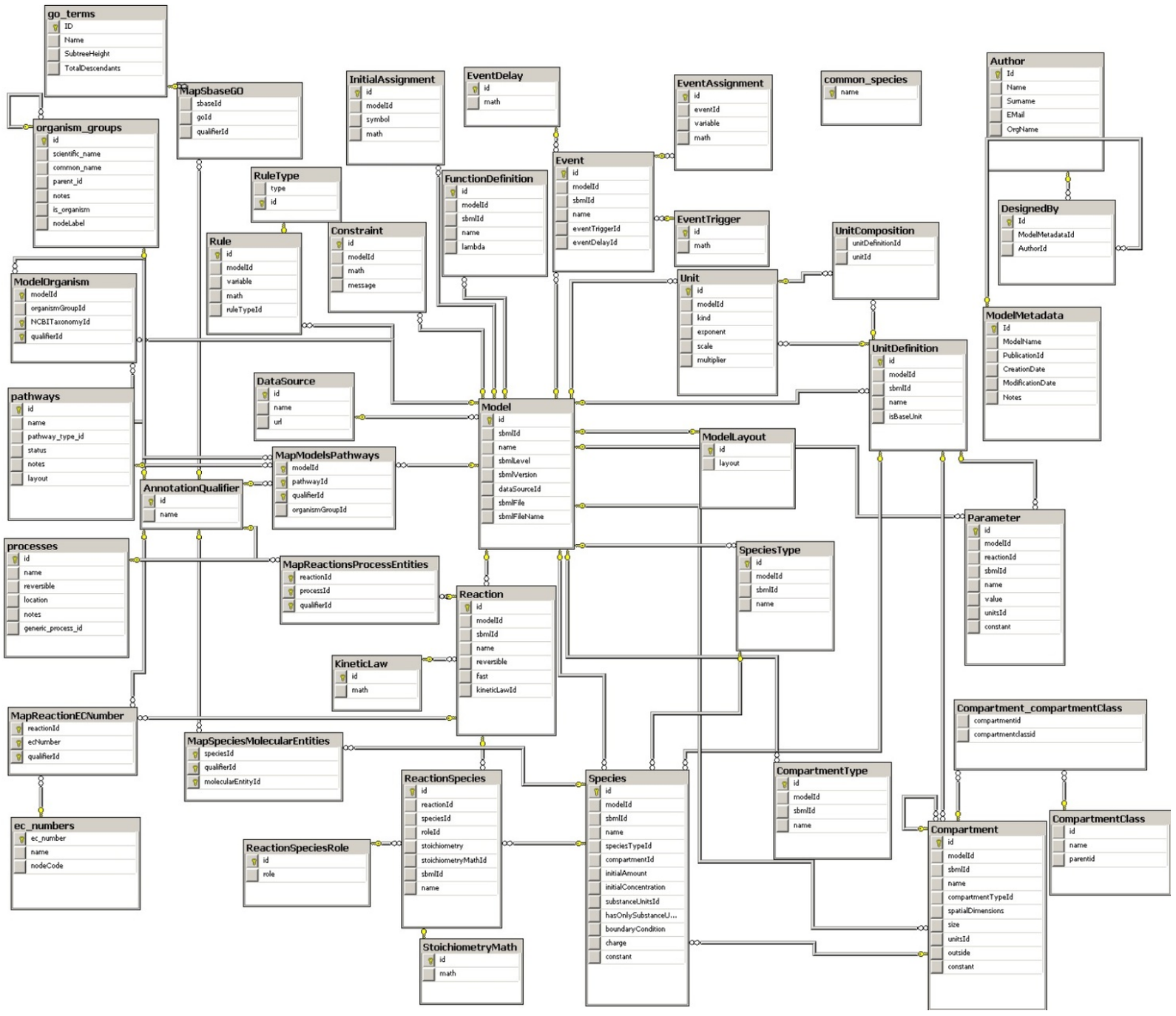
+90 Days

+1 Year

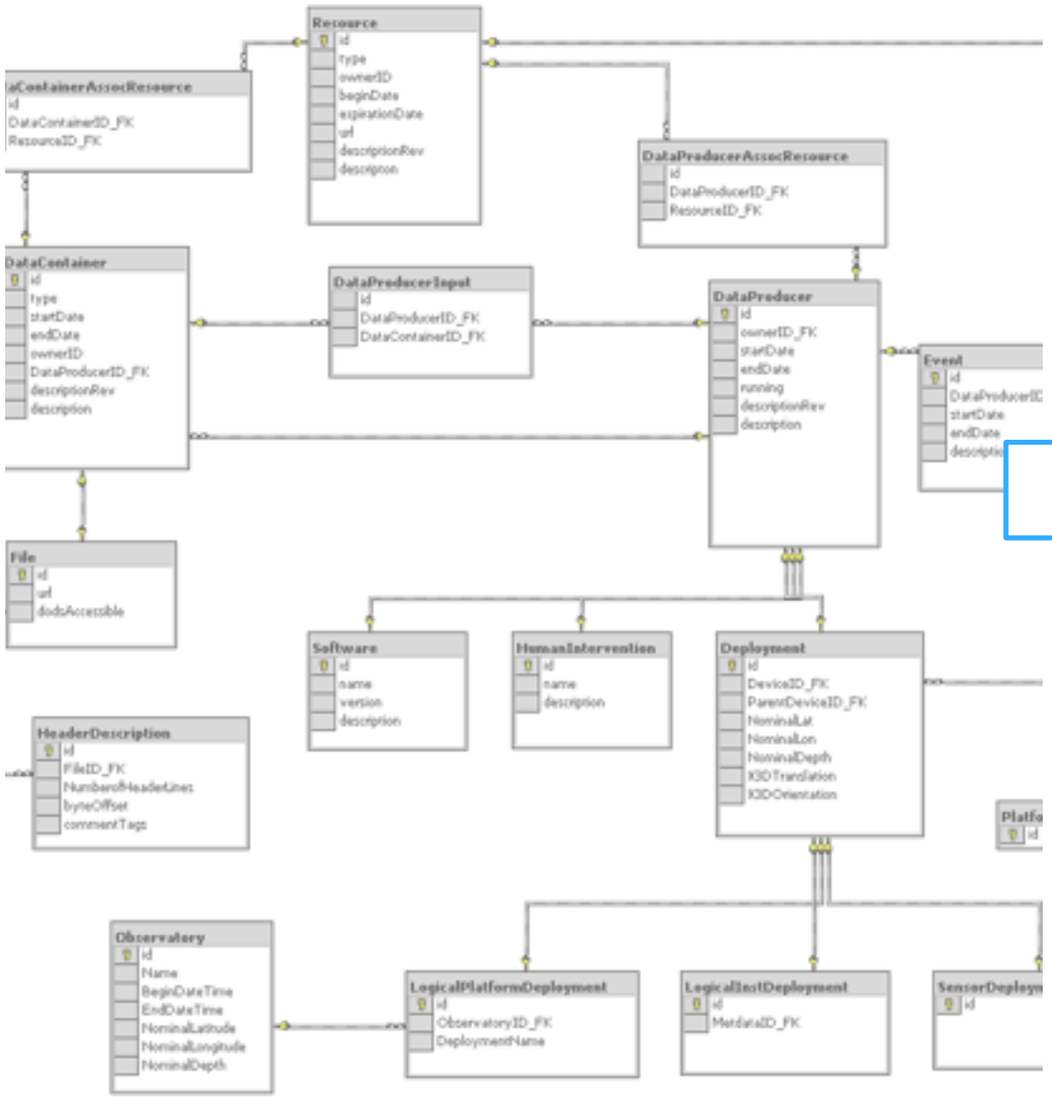
Productivity



Data models


go_terms
ID
Name
SubtreeHeight
TotalDescendants
MapSbaseGO
sbaseId
goId
qualifierId
organism_groups
id
scientific_name
common_name
parent_id
notes
is_organism
nodeLabel
InitialAssignment
id
modelId
symbol
math
EventDelay
id
math
EventAssignment
id
eventId
variable
math
common_species
name
Author
Id
Name
Surname
EMail
OrgName
RuleType
type
id
FunctionDefinition
id
modelId
sbmlId
name
lambda
Event
id
modelId
sbmlId
name
eventTriggerId
eventDelayId
EventTrigger
id
math
DesignedBy
Id
ModelMetadataId
AuthorId
Rule
id
modelId
variable
math
ruleTypeId
Constraint
id
modelId
math
message
UnitComposition
unitDefinitionId
unitId
ModelOrganism
modelId
organismGroupId
NCBITaxonomyId
qualifierId
Unit
id
modelId
kind
exponent
scale
multiplier
ModelMetadata
Id
ModelName
PublicationId
CreationDate
ModificationDate
Notes
UnitDefinition
id
modelId
sbmlId
name
isBaseUnit
pathways
id
name
pathway_type_id
status
notes
layout
DataSource
id
name
url
Model
id
sbmlId
name
sbmlLevel
sbmlVersion
dataSourceId
sbmlFile
sbmlFileName
ModelLayout
id
layout
MapModelsPathways
modelId
pathwayId
qualifierId
organismGroupId
AnnotationQualifier
id
name
Parameter
id
modelId
reactionId
sbmlId
name
value
unitsId
constant
processes
id
name
reversible
location
notes
generic_process_id
MapReactionsProcessEntities
reactionId
processId
qualifierId
SpeciesType
id
modelId
sbmlId
name
Reaction
id
modelId
sbmlId
name
reversible
fast
kineticLawId
KineticLaw
id
math
Compartment_compartmentClass
compartmentid
compartmentclassid
MapReactionECNumber
reactionId
ecNumber
qualifierId
MapSpeciesMolecularEntities
speciesId
qualifierId
molecularEntityId
ReactionSpecies
id
reactionId
speciesId
roleId
stoichiometry
stoichiometryMathId
sbmlId
name
Species
id
modelId
sbmlId
name
speciesTypeId
compartmentId
initialAmount
initialConcentration
substanceUnitsId
hasOnlySubstanceU...
boundaryCondition
charge
constant
CompartmentType
id
modelId
sbmlId
name
CompartmentClass
id
name
parentid
ec_numbers
ec_number
name
nodeCode
ReactionSpeciesRole
id
role
Compartment
id
modelId
sbmlId
name
compartmentTypeId
spatialDimensions
size
unitsId
outside
constant
StoichiometryMath
id
math

From tables to Documents




```
{
  title: ‘MongoDB’,
  contributors: [
    { name: ‘Eliot Horowitz’,
      email: ‘eh@10gen.com’ },
    { name: ‘Dwight Merriman’,
      email: ‘dm@10gen.com’ }
  ],
  model: {
    relational: false,
    awesome: true
  }
}
```

Documents

```
> p = {author: “roger”,
       date: new Date(),
       text: “about mongoDB...”,
       tags: [“tech”, “databases”]}

> db.posts.save(p)
```

Querying

```
> db.posts.find()

>    { _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
     author : "roger",
       date : "Sat Jul 24 2010 19:47:11",
     text : "About MongoDB...",
     tags : [ "tech", "databases" ] }
```

Secondary Indexes

```
//   1 means ascending, -1 means descending
> db.posts.ensureIndex({author: 1})
> db.posts.find({author: 'roger'})

>    { _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
     author : "roger",
      ... }
```

Conditional Query Operators

$all, $exists, $mod, $ne, $in, $nin, $nor,
$or, $size, $type, $lt, $lte, $gt, $gte

```
// find posts with any tags
> db.posts.find( {tags: {$exists: true }} )

// find posts matching a regular expression
> db.posts.find( {author: /^rog*/i } )

// count posts by author
> db.posts.find( {author: 'roger'} ).count()
```

Atomic Updates

$set, $unset, $inc, $push, $pushAll, $pull, $pullAll, $bit

```
> comment = { author: “fred”,
              date: new Date(),
              text: “Best Movie Ever”}

> db.posts.update( { _id: “...” },
              $push: {comments: comment} );
```

Nested Documents

```
{  _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
   author : "roger",
   date : "Sat Apr 24 2011 19:47:11",
   text : "About MongoDB...",
   tags : [ "tech", "databases" ],
   comments : [
          {
             author : "Fred",
             date : "Sat Apr 25 2010 20:51:03 GMT-0700",
             text : "Best Post Ever!"
          }
      ]
}
```

Advanced indexing

```
// Index nested documents
> db.posts.ensureIndex( “comments.author”: 1)
> db.posts.find({‘comments.author’:’Fred’})

// Index on tags (multi-key index)
> db.posts.ensureIndex( tags: 1)
> db.posts.find( { tags: ‘tech’ } )

// geospatial index
> db.posts.ensureIndex( “author.location”: “2d” )
> db.posts.find( “author.location”: { $near : [22,42] } )
```

Rich Documents

```
{    _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),

     line_items : [ { sku: ‘tt-123’,
                      name: ‘Coltrane: Impressions’ },
                    { sku: ‘tt-457’,
                      name: ‘Davis: Kind of Blue’ } ],

     address : { name: ‘Banker’,
                 street: ‘111 Main’,
                 zip: 10010 },

     payment: { cc: 4567,
                exp: Date(2011, 7, 7) },

     subtotal: 2355
}
```

springone 2GX CHICAGO 2011

Demo

Cloud Foundry and Mongo DB

springsource
A division of vmware

Spring Data Intoduction

What is Spring Data all about?

- Bring classic Spring value propositions to NOSQL
  - Productivity
  - Programming model consistency
- Support for a wide range of NOSQL databases
- Also, support for new features for JDBC and JPA
- Support for other data related projects like Hadoop and Gemfire

Spring Data MongoDB Support

Spring Data support for MongoDB:

- MongoTemplate
  - ✓MongoConverter interface for mapping Mongo documents
    - Built-in Advanced Mapping
      - – Annotation based (@Document, @Id, @DbRef)
    - MappingMongoConverter for POJO mapping support
      - Leverage Spring 3.0 TypeConverters and SpEL
  - ✓Exception translation
  - ✓Java based Query, Criteria, and Update DSLs
- MongoRepository
  - ✓Built on Spring Data JPA (Hades) support for JPA Repositories
  - ✓QueryDSL integration to support type-safe queries.

Working with POJOs

```
@Document
public class Book {

     private String title;

     @DBRef
     private Author author;

     @Id
     private String isbn;

     private BigDecimal price;

     private Date published;

     private Set<String> categories;

     // getters and setters
}
```

```
@Document
public class Author {

     @Id @SuppressWarnings("unused")
     private String id;

     private String name;

     public String getName() {
          return name;
     }

     public void setName(String name) {
          this.name = name;
     }
}
```

Mongo Template CRUD Methods

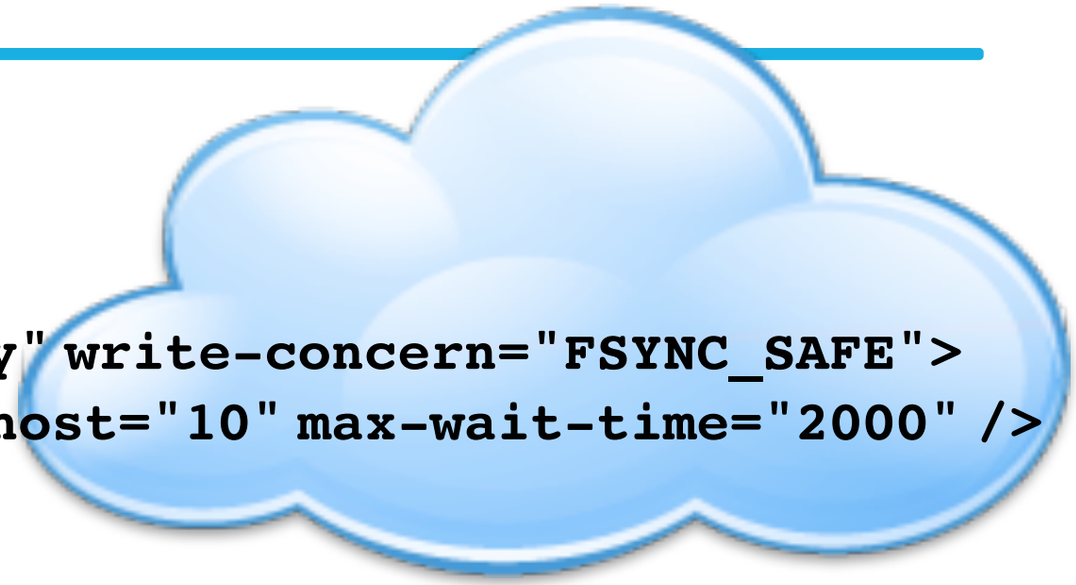
```
Mongo mongo = new Mongo("localhost", 27017);
MongoDbFactory mongoDbFactory =
      new SimpleMongoDbFactory(mongo, "db");
MongoTemplate mongoTemplate = new
MongoTemplate(mongoDbFactory);


mongoTemplate.insert(book);

List<Book> books = mongoTemplate.findAll(Book.class);

Book book = mongoTemplate.findOne(query, Book.class);

mongoTemplate.save(book);

mongoTemplate.remove(query, Book.class);
```

Querying

```
public List<Book> findByCategoriesOrYear(Set<String> categories, String year) {
      String[] categoriesToMatch;
      if (categories == null) {
            categoriesToMatch = new String[] {};
      }
      else {
            categoriesToMatch = categories.toArray(new String[categories.size()]);
      }
      Date startDate = null;
      if (year != null && year.length() == 4) {
            DateFormat formatter = new SimpleDateFormat("yyyy-dd-MM");
            try {
                  startDate = formatter.parse(year + "-01-01");
            } catch (ParseException e) {}
      }
      Criteria searchCriteria = null;
      if (startDate != null) {
          searchCriteria =
              Criteria.where("published").gte(startDate);
      }
      else {
            searchCriteria = new Criteria();
      }
      if (categoriesToMatch.length > 0) {
          searchCriteria
              .and("categories").in((Object[])categoriesToMatch);
      }
      Query query = new Query(searchCriteria);
      return mongoTemplate.find(query, Book.class);
}
```

Cloud Foundry MongoDB Support

Namespace support for MongoDB



```
<cloud:mongo-db-factory id="mongoDbFactory" write-concern="FSYNC_SAFE">
    <cloud:mongo-options connections-per-host="10" max-wait-time="2000" />
</cloud:mongo-db-factory>
```

```
<dependency>
    <groupId>org.cloudfoundry</groupId>
    <artifactId>cloudfoundry-runtime</artifactId>
    <version>0.8.1</version>
</dependency>

<repository>
    <id>org.springframework.maven.milestone</id>
    <name>Spring Framework Maven Milestone Repository</name>
    <url>http://maven.springframework.org/milestone</url>
</repository>
```

Demo

MongoTemplate example:

https://github.com/trisberg/springone-mongotemplate

Spring Data Repository

Spring Data Repository basics:

- Generic repository implementation
- Basic CRUD (create, read, update and delete) methods
- Generating code for queries defined in repository interface
  - findAll
  - findByName ...
- Pagination and sorting support
- Currently has JPA and Mongo implementations

Spring Data Repository Interfaces

CrudRepository

long	**count**() Returns the number of entities available.
void	**delete**(ID id) Deletes the entity with the given id.
void	**delete**(Iterable<? extends T> entities) Deletes the given entities.
void	**delete**(T entity) Deletes a given entity.
void	**deleteAll**() Deletes all entities managed by the repository.
boolean	**exists**(ID id) Returns whether an entity with the given id exists.
Iterable<T>	**findAll**() Returns all instances of the type.
T	**findOne**(ID id) Retrives an entity by its primary key.
Iterable<T>	**save**(Iterable<? extends T> entities) Saves all given entities.
T	**save**(T entity) Saves a given entity.

PagingAndSortingRepository

Page<T>	**findAll**(Pageable pageable) Returns a Page of entities meeting the paging restriction provided in the Pageable object.
Iterable<T>	**findAll**(Sort sort) Returns all entities sorted by the given options.

Spring Data Repository Query Methods

Spring Data Repository query methods:

Keyword	Sample	Logical result
GreaterThan	findByAgeGreaterThan(int age)	{"age" : {"$gt" : age}}
LessThan	findByAgeLessThan(int age)	{"age" : {"$lt" : age}}
Between	findByAgeBetween(int from, int to)	{"age" : {"$gt" : from, "$lt" : to}}
IsNotNull, NotNull	findByNameNotNull()	{"name" : {"$ne" : null}}
IsNull, Null	findByNameNull()	{"name" : null}
Like	findByNameLike(String expr)	{"name" : expr} (expr as regex)
(No keyword)	findByName(String name)	{"name" : name}
Not	findByNameNot(String name)	{"name" : {"$ne" : name}}

and more, see reference docs ...

Mongo Repository Example

```
public interface BookRepository extends Repository<Book, String> {

    Book save(Book book);

    Book findOne(String isbn);

    void delete(String isbn);

    List<Book> findAll();

    List<Book> findByPublishedGreaterThan(Date date);

    List<Book> findByCategoriesIn(String[] categories);

    List<Book> findByPublishedGreaterThanAndCategoriesIn(
                    Date date, String[] categories);

}
```

Demo

Mongo Repository example:

https://github.com/trisberg/springone-repository

QueryDSL Repository

QueryDSL allows you to build typesafe queries using Java for JPA, SQL, MongoDB and more

- Generic repository implementation
- Code completion in IDE
- No Strings
  - reference types and properties safely
  - easier refactoring
- Incremental query definition is easier
- Annotation preprocessor generates query types

Demo

Mongo QueryDSL Repository example:

https://github.com/trisberg/springone-repo-qdsl

Why run MongoDB in the cloud?

CLOUD FOUNDRY

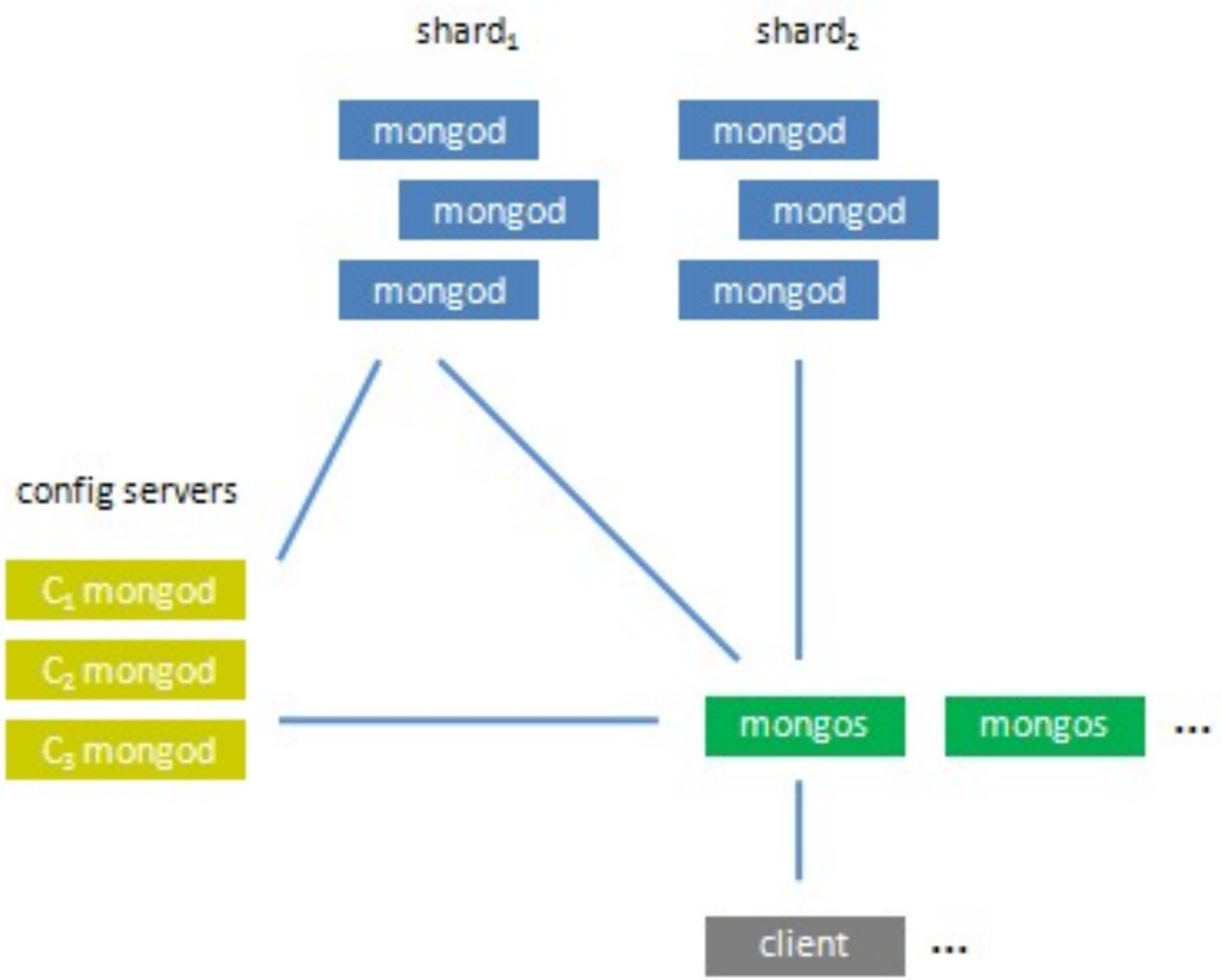
+

mongoDB

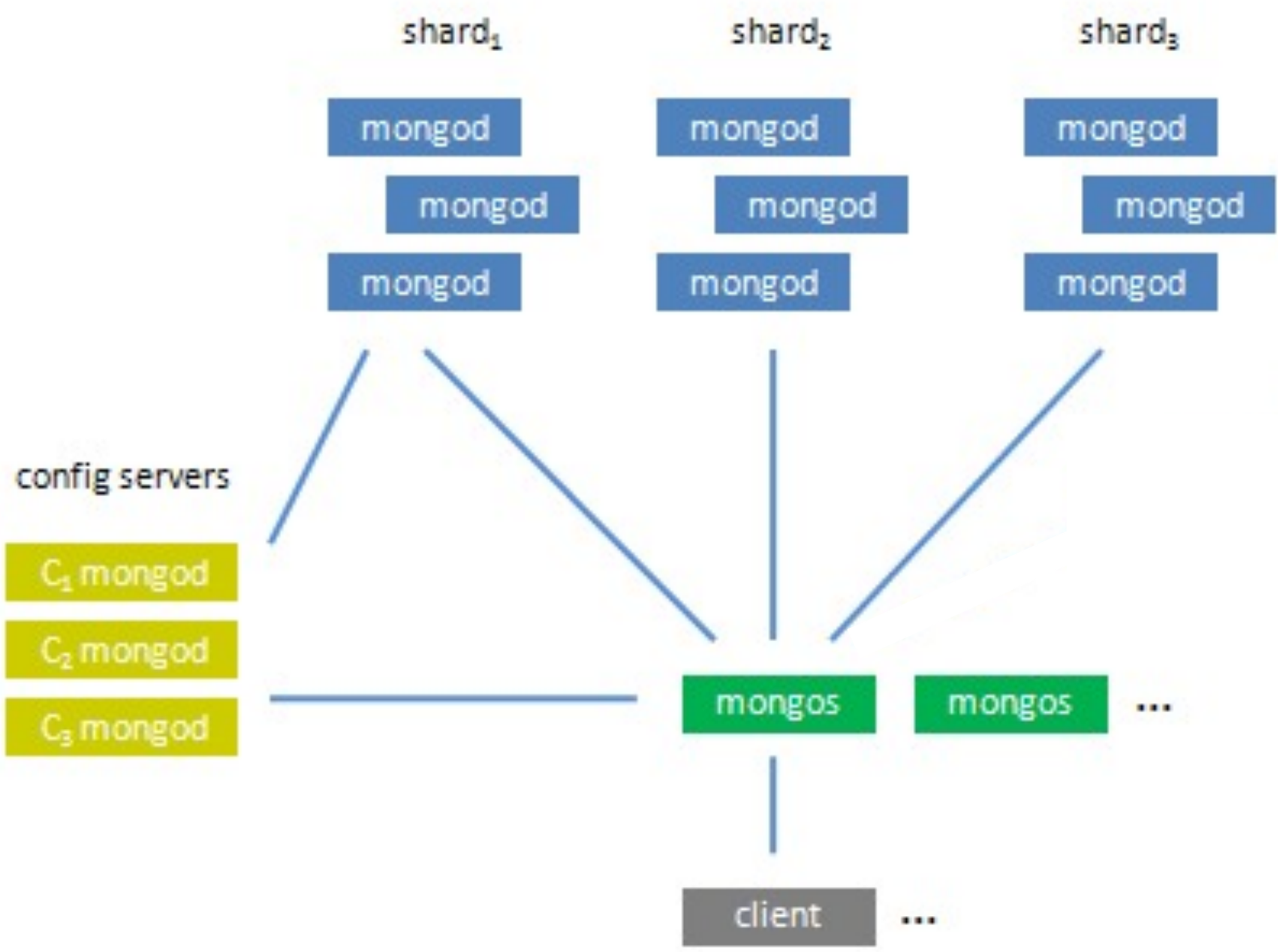
Typical scaling approaches don't work

MongoDB scales out

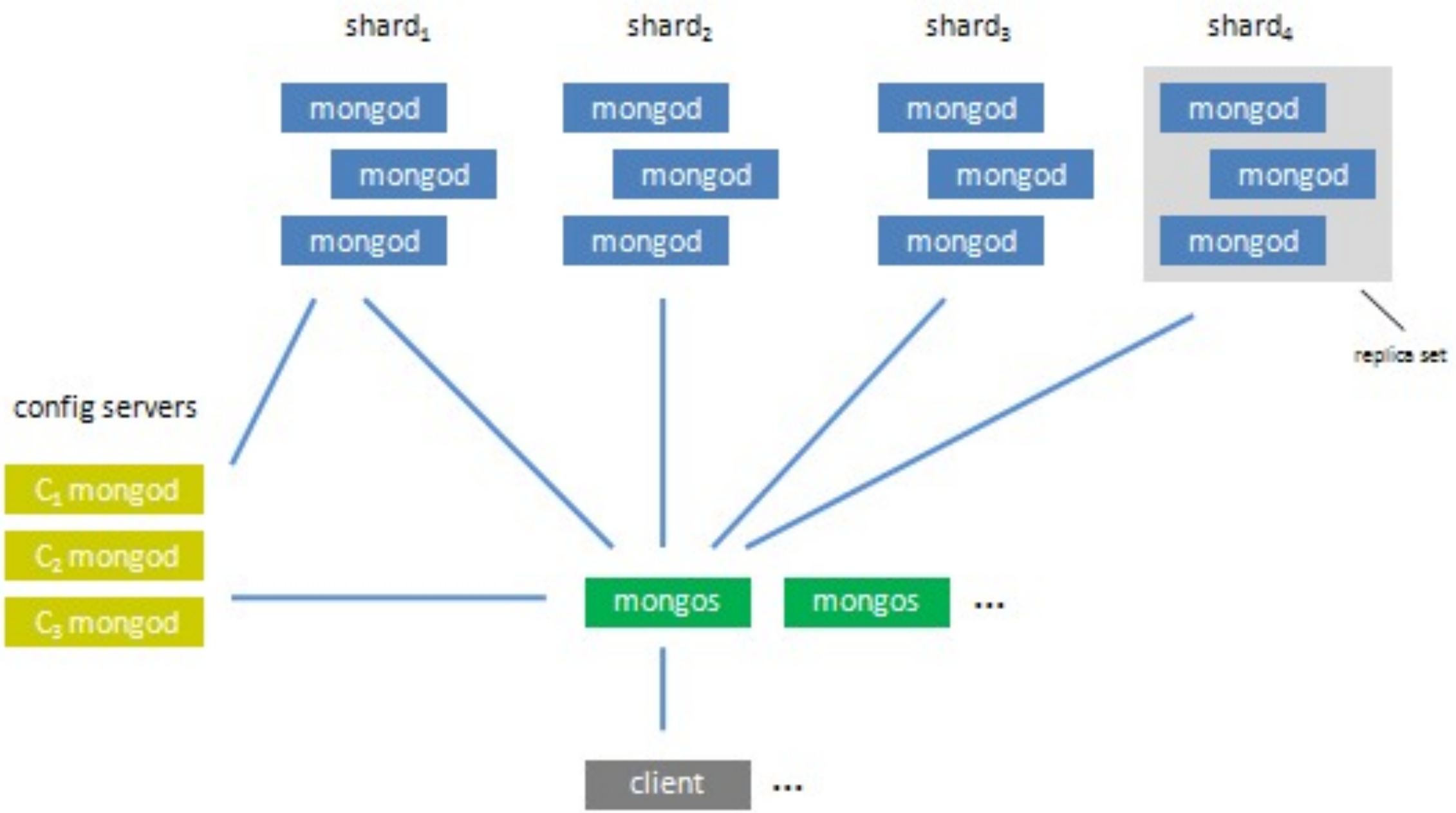
MongoDB scales out



MongoDB scales out



MongoDB scales out



MongoDB in the cloud

- Elastic database capacity
  - Add more storage, compute, memory dynamically
- Portable across environments
  - Public cloud
  - Private cloud
  - Micro cloud
- Simplifies operations
  - No need to engineer solutions for one-off hardware platforms
  - “Just another VM”

springone 2GX CHICAGO 2011

Q&A

springsource
A division of vmware