Transazioni MongoDB

Atomicità delle transazioni in un Mongo-Cluster

Progetto per il corso di New Generation Databases

Baioni Francesco

Caprari David

- Cluster MongoDB
  - Configurazioni dei nodi
  - Script inizializzazione
- Applicazione interfaccia grafica
  - Semplice GUI (WinUI)
  - Backend con interfaccia al driver

Cluster MongoDB

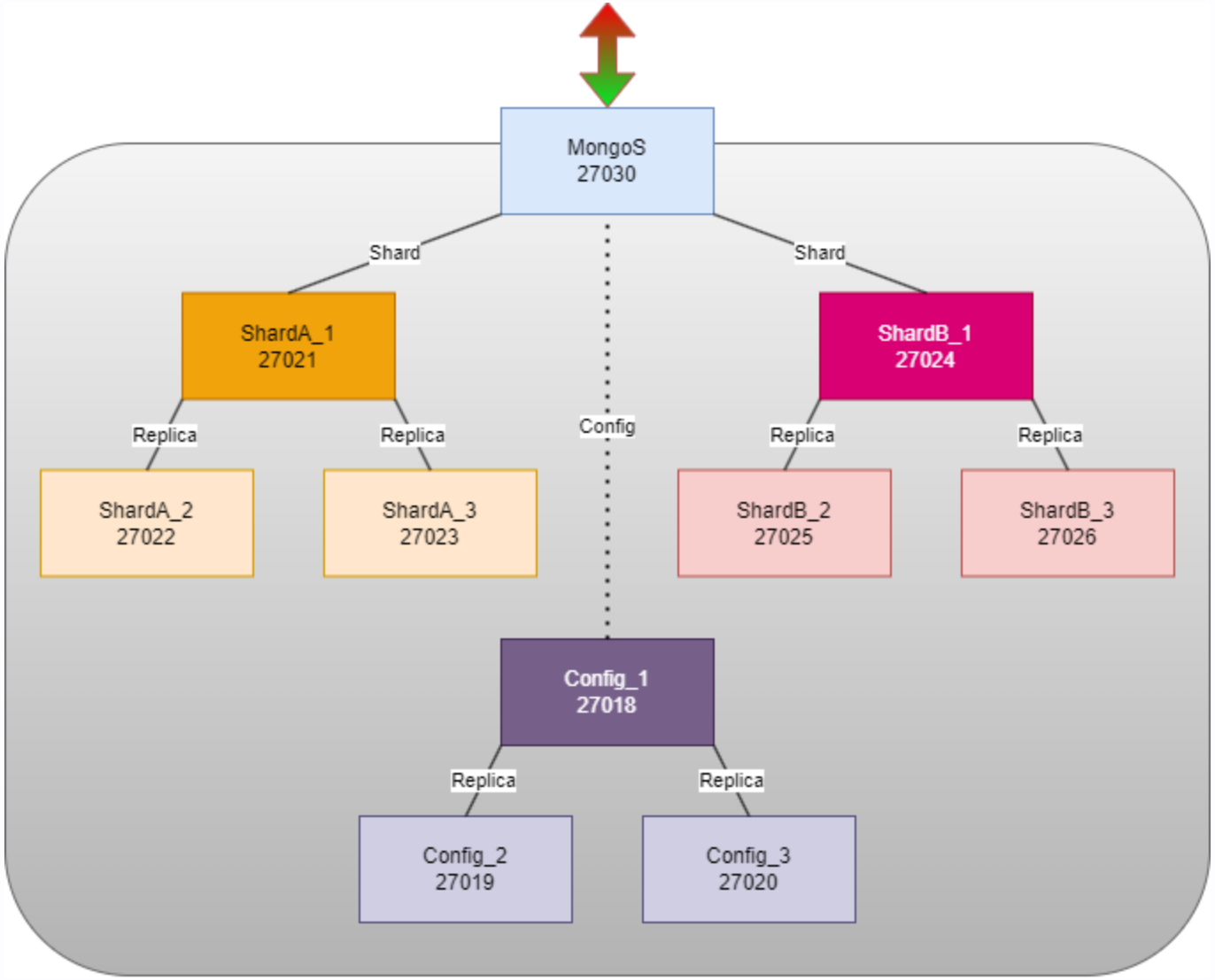
- MongoDB 6.0.2
- MongoS 6.0.2
- MongoShell

Struttura interna

- Shard A
- Shard B

Ogni shard composto da 3 Replica Set
In locale (localhost:porta)

Struttura interna



Dati

Singola collezione di due documenti

Documento **Y** --> Shard A

Documento **X** --> Shard B

```
{
    "_id": ObjectId("..."),
    "name": "x",
    "value": "10",
    "shard_value": "0",
    "lastModified": "ISODate("aaaa-mm-ggT:...")"
}
```

Applicazione WinUI

- VisualStudio2022
- Framework XAML/C# WinUI3

Interfaccia al DB

Funzioni base:

- Inserimento
- Ricerca
- Aggregazione
- Cancellazione

Funzioni avanzate:

- **Replica Update**
- **Sharded Update**

Product Catalog

Inserimento

Insert { "name" : "z" }

Find { "field" : "value" }

Aggregate { $match: { country: "South Korea" }}

Delete { "field" : "value" }

Replica Update { "name" : "x" } { "value" : "10" }

Sharded Update { "value" : "1000" }

Inserted

Updated

Product Catalog

Ricerca

Insert { "name" : "z" }

Find { "name" : "z" }

Aggregate { $match: { country: "South Korea" }}

Delete { "field" : "value" }

Replica Update { "name" : "x" } { "value" : "10" }

Sharded Update { "value" : "1000" }

{ "_id" : ObjectId("6436e0af862ea2a970b15858"), "name" : "z" }

Updated

Product Catalog

Aggregazione

Insert { "name" : "z" }

Find { "name" : "z" }

Aggregate { $match: { name: "x" }}

Delete { "field" : "value" }

Replica Update { "name" : "x" } { "value" : "10" }

Sharded Update { "value" : "1000" }

{ "_id" : ObjectId("642431c5d8576e4a7ef23b24"), "name" : "x", "value" : "1000", "shard_value" : "0", "lastModified" : ISODate("2023-04-12T16:47:03.135Z") }

Updated

Product Catalog

Cancellazione

Insert { "name" : "z" }

Find { "name" : "z" }

Aggregate { $match: { name: "x" }}

Delete { "name" : "z" }

Replica Update { "name" : "x" } { "value" : "10" }

Sharded Update { "value" : "1000" }

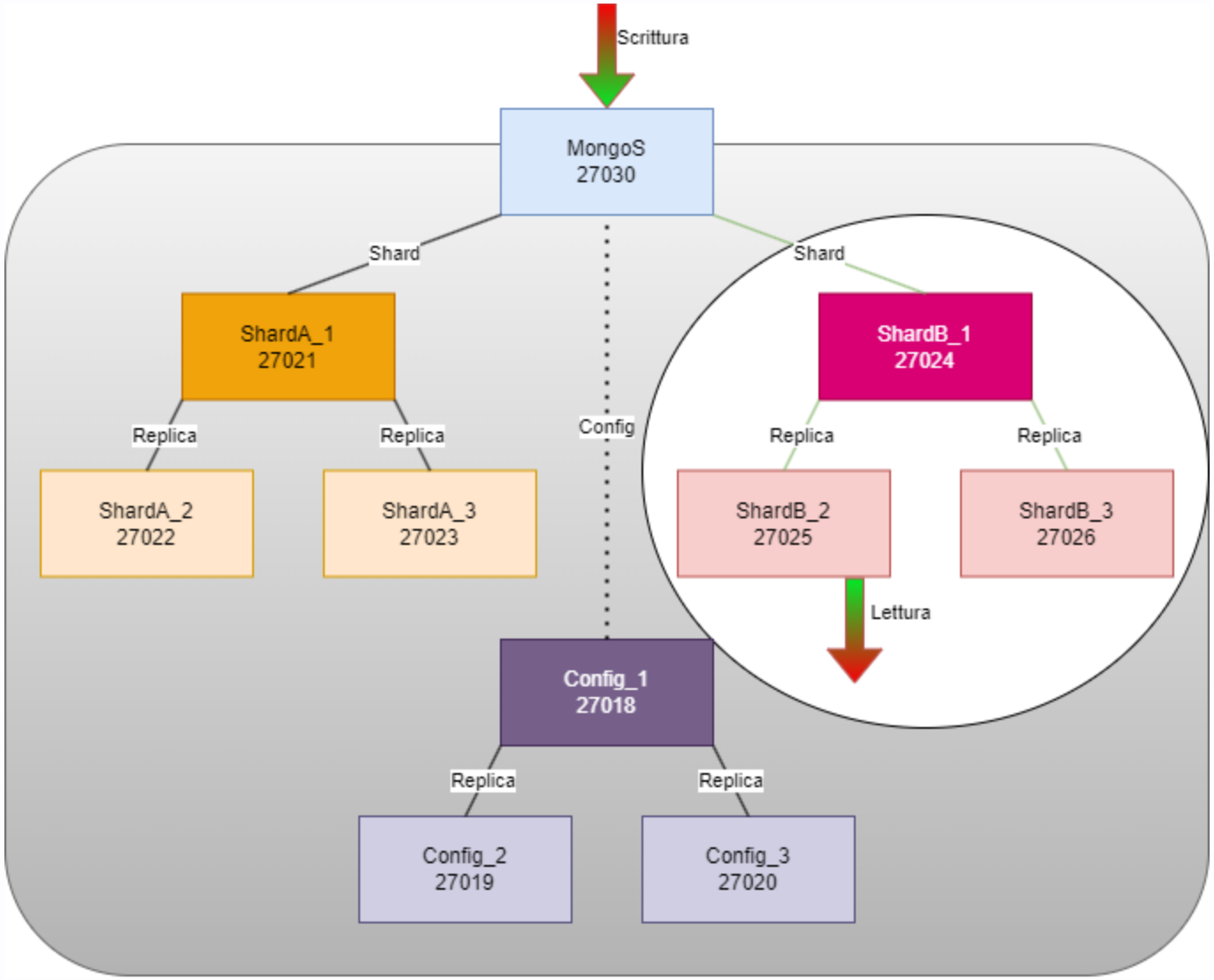
Deleted

Updated

Replica Update

- Aggiornamento asincrono attraverso MongoS (localhost:27030) del "value" nel documento X
- Contemporaneo check di consistenza dell'aggiornamento attraverso ReplicaSet secondario (localhost:27025)

Replica Update



Product Catalog

Replica Update

Insert { "field" : "value" }

Find { "field" : "value" }

Aggregate { $match: { country: "South Korea" }}

Delete { "field" : "value" }

Replica Update { "name" : "x" } { "value" : "10" }

Sharded Update { "value" : "1000" }

{ "_id" : ObjectId("642431c5d8576e4a7ef23b24"), "name" : "x", "value" : "10", "shard_value" : "0", "lastModified" : ISODate("2023-04-12T16:45:48.574Z") }
Updated in 53 milliseconds, with 0 inconsistency checks.

Updated

Replica Update

Replica delle transazioni interne allo shard è atomica

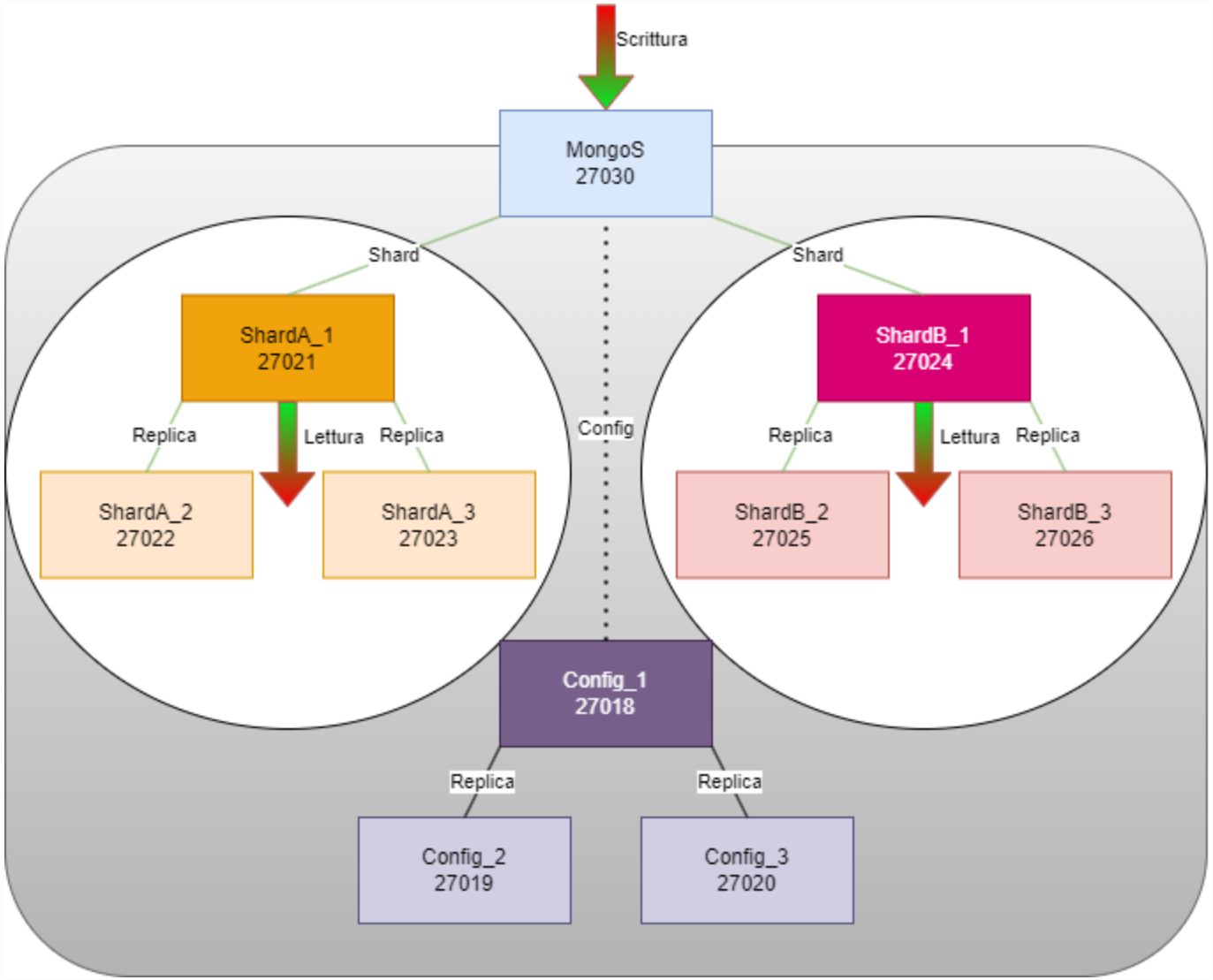
MongoDB $\geq$ 4.2 supporta Multi-Document transactions

Sharded Update

- **Transazione asincrona a MongoS:**
  1. Aggiornamento del "value" nel documento Y
  2. Attesa di 1sec
  3. Aggiornamento del "value" nel documento X
- **Check contemporaneo di vincolo di integrità**

  "value@X"=="value@Y"

Sharded Update



Product Catalog

Sharded Update

Insert { "field" : "value" }

Find { "field" : "value" }

Aggregate { $match: { country: "South Korea" }}

Delete { "field" : "value" }

Replica Update { "name" : "x" } { "value" : "10" }

Sharded Update { "value" : "1000" }

Updated in 1011 milliseconds, with 10 inconsistency checks.

Updated

Sharded Update

Transazione multi-shard NON è atomica

Le transazioni che coinvolgono più di uno shard non sono atomiche

Grazie per l'attenzione

Post-Notes:

1. Bug multi-istanza nello stesso host
2. Codice Replica Update
3. Codice Sharded Update

Replica Update

```
private async void ReplicaUpdateButton_Click(object sender, RoutedEventArgs e)
    {
        ...
        using (var session = GetConnectionClient().StartSession())
        {
            ...
            IMongoCollection<BsonDocument> collection_sharded = database_sharded.GetCollection<BsonDocument>("sharded_coll");
            var filter = Builders<BsonDocument>.Filter.Eq(ToUpdateTextBox.Text.Split("\"")[1], ToUpdateTextBox.Text.Split("\"")[3]);
            var resFilter = collection.Find(filter).FirstOrDefault();
            if (resFilter != null)
            {
                ...
                var update = Builders<BsonDocument>.Update.Set(new_key, new_value).CurrentDate("lastModified");
                try
                {
                    await collection.UpdateManyAsync(filter, update);
                    connectionBlock.Text = "Updated";
                    var checks = 0;
                    DateTime now = DateTime.Now;
                    bool checked_replica_copy = false;
                    try
                    {
                        while (!checked_replica_copy && checks < 1000)
                        {
                            var resDocument = collection_sharded.Find(filter).FirstOrDefault();
                            if (resDocument != null)
                            {
                                if (resDocument[new_key].Equals(new_value))
                                {
                                    checked_replica_copy = true;
                                    DateTime new_now = DateTime.Now;
                                    now = resDocument["lastModified"].ToLocalTime();
                                    ResultTextBlock.Text = resDocument.ToString() + /
                                                           "\nUpdated in " + (new_now.Millisecond + new_now.Second * 1000 - now.Millisecond - now.Second * 1000).ToString() /
                                                           + " milliseconds, with " + checks.ToString() /
                                                           + " inconsistency checks.";
                                }
                                else { checks++; }
                            }
                        ...
                        }
                        if (checks >= 1000)
                        {
                            ResultTextBlock.Text = "More than 1000 inconsistency checks. Are you on the correct shard? (Actually on " + new_connection_string + ").";
                        }
        }
    }
```

Sharded Update - ext

```
private async void ShardedUpdateButton_Click(object sender, RoutedEventArgs e)
    {
        ...
        using (var session = GetConnectionClient().StartSession())
        {
            ...
            var new_key = ShardedUpdatedTextBox.Text.Split("\"")[1];
            var new_value = ShardedUpdatedTextBox.Text.Split("\"")[3];-
            if (resFilterX != null && resFilterY != null)
            {
                DateTime now = DateTime.Now;
                session.StartTransaction();
                ShardedUpdateAsync(collection, new_key, new_value);
                var checks = 0;
                int num_checks = 100; //Number of consistency checks scheduled
                while (checks < num_checks)
                {
                    var x = collection_shardB.Find(filterX).First();
                    var y = collection_shardA.Find(filterY).First();
                    if (y["value"] == x["value"])
                    {
                        DateTime new_now = x["lastModified"].ToLocalTime();
                        ResultTextBlock.Text = "Updated in " /
                        + (new_now.Millisecond + new_now.Second * 1000 - now.Millisecond - now.Second * 1000).ToString() /
                        + " milliseconds, with " + checks.ToString() + " inconsistency checks.";
                        break;
                    }
                    else
                    {
                        checks++;
                        await Task.Delay(100);
                    }
                }
                if (checks >= num_checks)
                {    ResultTextBlock.Text = "Check resulted in more than " + checks.ToString() + " consistency checks failed.";    }
            }
        ...
        }
    }
```

Sharded Update - transaction

```
private async Task ShardedUpdateAsync(IMongoCollection<BsonDocument> collection, string new_key, string new_value)
    {

        collection.UpdateMany(Builders<BsonDocument>.Filter.Eq("name", "y"), /
                                                    Builders<BsonDocument>.Update.Set(new_key, new_value)/
                                                    .CurrentDate("lastModified"));
        await Task.Delay(1000);
        collection.UpdateMany(Builders<BsonDocument>.Filter.Eq("name", "x"), /
                                                    Builders<BsonDocument>.Update.Set(new_key, new_value)/
                                                    .CurrentDate("lastModified"));
        connectionBlock.Text = "Updated";
    }
```