CatchAndRetry

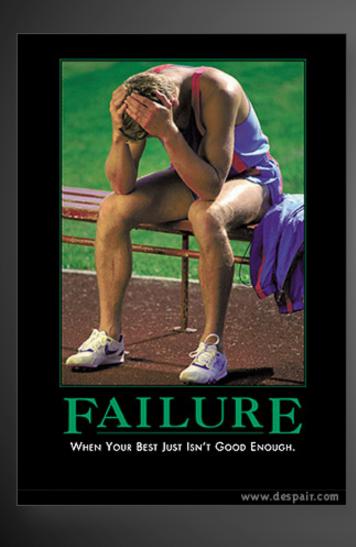
Extending Exceptions to Handle Distributed Systems Failure and Recovery

Emre Kıcıman, Ben Livshits, and Madanlal Musuvathi

Microsoft Research

Fluxo Project

Motivation: Failures in Distributed Systems



- Failures are a fact of life
- Especially in distributed systems
- Developer has to deal with that possibility
- CatchAndRetry make error recovery easier

Availability Errors

- Network disconnects
- Hardware failures
- Performance stutters



Data Staleness and Consistency



How to Recover?

- Ignore and continue optimistically
- Retry (N times)
- Retry with different parameters
- Schedule the retry for a later time

CatchAndRetry Illustrated



Basic retry



Retry Multiple Times



Handle Retry Failure



Recovery from Multiple Exceptions

```
try {
      ReadData();
} catch(NetworkConnectivityException e) {
      Thread.Sleep(TimeSpan.FromMilliseconds(5));
      retry 5;
} fail {
      Console.Writeline("network failure");
} catch(StaleDataException e) {
      RefreshData();
      retry 10;
```



Parametrized Retry

```
int x = 10, y = 100, z = 200;
try(x,y) {
    z = x + y;
} catch(Exception ex) {
    // retry twice with x=5
    retry(x=5), 2;
}
```



Re-scheduled Retry

```
try {
    // do something
} catch(
    AvailabilityException e,
    RetryFunction r)
    // try block asynchronously
    scheduleForLater(r);
```

De-sugaring

```
A a;
try(a){
           T;
} catch(Exception1 ex1){
           P1;
           retry 3;
           Q1;
} fail {
           F1;
} catch(Exception2 ex2){
           P2;
           retry(a=a2);
           Q2;
} fail {
           F2;
} finally{
           R;
```

- Try block becomes a parameterized delegate
- Every retry statements leads to a *retry counter*
 - Possible to alternate between retries
 - Gives an upper bound on the number of retries

Facebook Examples

Getting a Friend List in FB (staleness)

```
try {
   List<Person> f = friends.Get(person, staleness);
} catch (StalenessException ex){
   Aspects.registerAspect(Facebook.PingServer,
     new Task(delegate(){
       // refresh from server
       friends.Refresh(person);
   })).waitForCompletion();
   // just retry
   Console.WriteLine("Retryng after update...");
   retry;
```

Server Availability

```
string protocol="HTTP";
try(protocol){
   SendMessage(friend, message, protocol);
} catch(AvailablityException ex){
   Thread.Sleep(TimeSpan.FromSeconds(5));
   retry(protocol="HTTP");
}
```

CatchAndRetry Summary

- Extensions of traditional try/catch/finally
- Focus on common types of distributed systems errors
 - Availability
 - Consistency
- Present our vision for extending C#/Java/JavaScript
- But also more experimental systems (Fluxo)