Fault Tolerance in akka

4010-441
Principles of Concurrent Software Systems

Akka proponents believe that Actors naturally form hierarchies.

- Within the hierarchy there is a strong Supervision relationship
  - ***Parental supervision***
  - ***Only actors create actors***
  - ***Each actor is supervised by its parent***
- Upon failure, subordinate actor will
  - ***Suspend itself, and its subordinates***
  - ***Send message to supervisor indicating the failure***

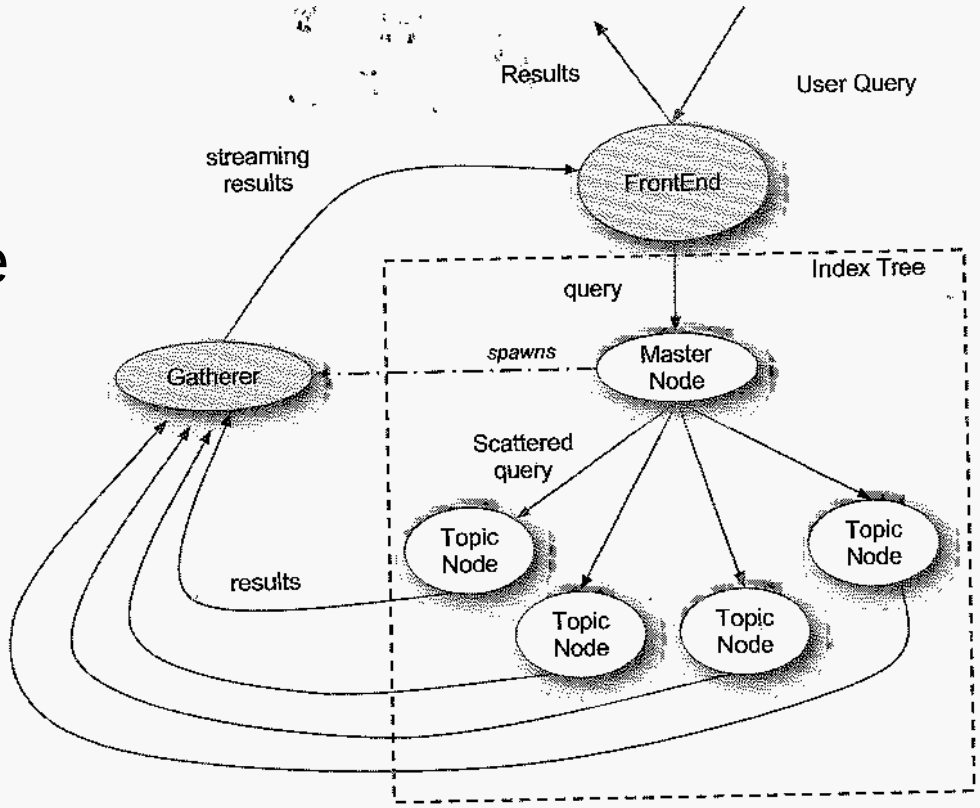
Material on akka from on-line documentation
http://doc.akka.io/docs/akka/snapshot/java/FaultTolerance.html

“Let it Fail” (“Let it Crash”) Strategy

- Popularized by Erlang
- In contrast to try-catch exception handling
  - ***Instead of trying all things possible to prevent an error from happening, this approach embraces failure***
- Let actors fail and let supervisor actor handle failure
  - ***Restart a failed actor***
  - ***Restart a portion of the system it supervises***
  - ***Pass failure to its supervisor***
- Failure is isolated and prevented from affecting other parts of the system
  - ***Failure Zones***

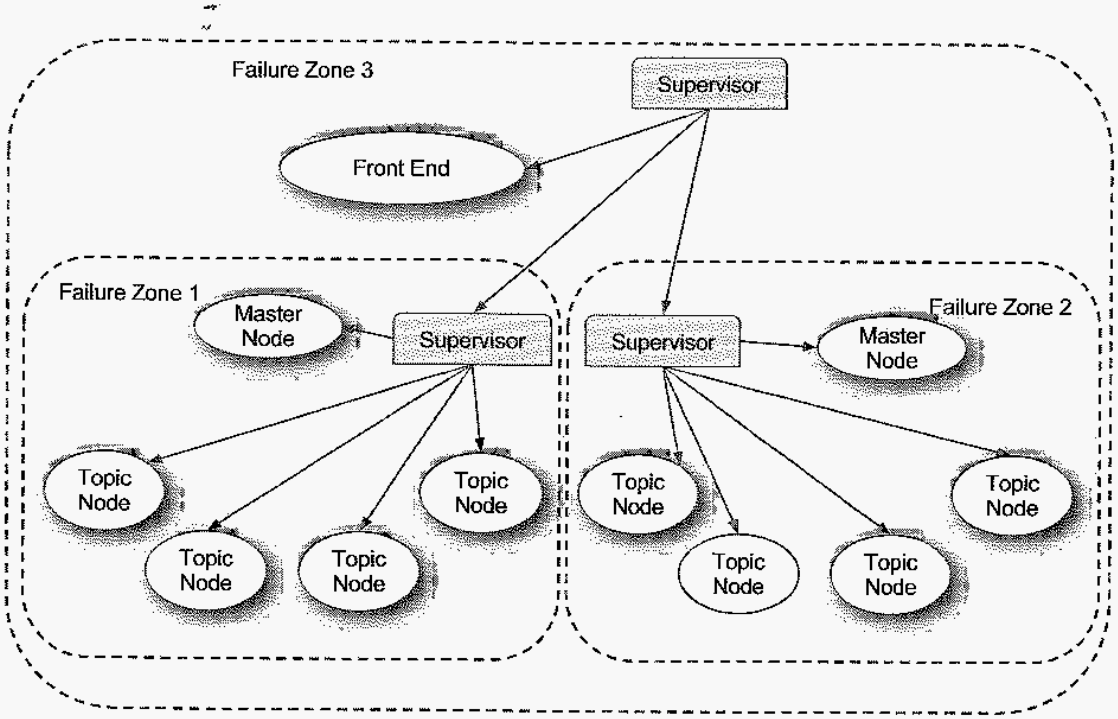
Failure Zones

- Scatter-Gather Search
- Master Node actor(s) is supervisor of Topic Node actors
- Design zones to allow master node or topic nodes to fail without bringing down the entire system



From "Scala in Depth", Joshua Suereth

Scatter-Gather Failure Zones



- Failure Zones 1 & 2 supervisors are responsible for starting the entire tree or one particular topic node
- Failure Zone 3 manages the search on the front end. It restarts the underlying search trees or front end as needed

From "Scala in Depth", Joshua Suereth

A supervisor actor has a range of options for handling the failure notice.

- Resume the subordinate and all its subordinates, keep supervisor state
- Restart the subordinate and all its subordinates, clear supervisor state
- Terminate the subordinate permanently
- Escalate the failure.

Each supervisor needs a function which translates all failure causes into one of the above options.

Actor Restart Strategies

- Superviosr Actors have two different restart strategies:
  - ***OneForOne: Restart only the component that has crashed.***
  - ***AllForOne: Restart all the components that the supervisor is managing, including the one that have crashed.***

An example supervisor strategy

```
private static SupervisorStrategy strategy =
    new OneForOneStrategy(10, Duration.create("1 minute"),
    new Function<Throwable, Directive>() {

    @Override
    public Directive apply(Throwable t) {
        if (t instanceof ArithmeticException) {
            return resume();
        } else if (t instanceof NullPointerException) {
            return restart();
        } else if (t instanceof IllegalArgumentException) {
            return stop();
        } else {
            return escalate();
        }
    }
});

@Override
public SupervisorStrategy supervisorStrategy() {
    return strategy;
}
```

Maximum number of restarts for actor within the time range. No more than 10 restarts within 1 minute.

The causes of failures can be categorized three ways.

- Systematic error for received message
- Transient failure of an external resource
- Corrupt internal state

If a supervisor believes that neither itself nor its other children are affected, the child can simply be restarted.

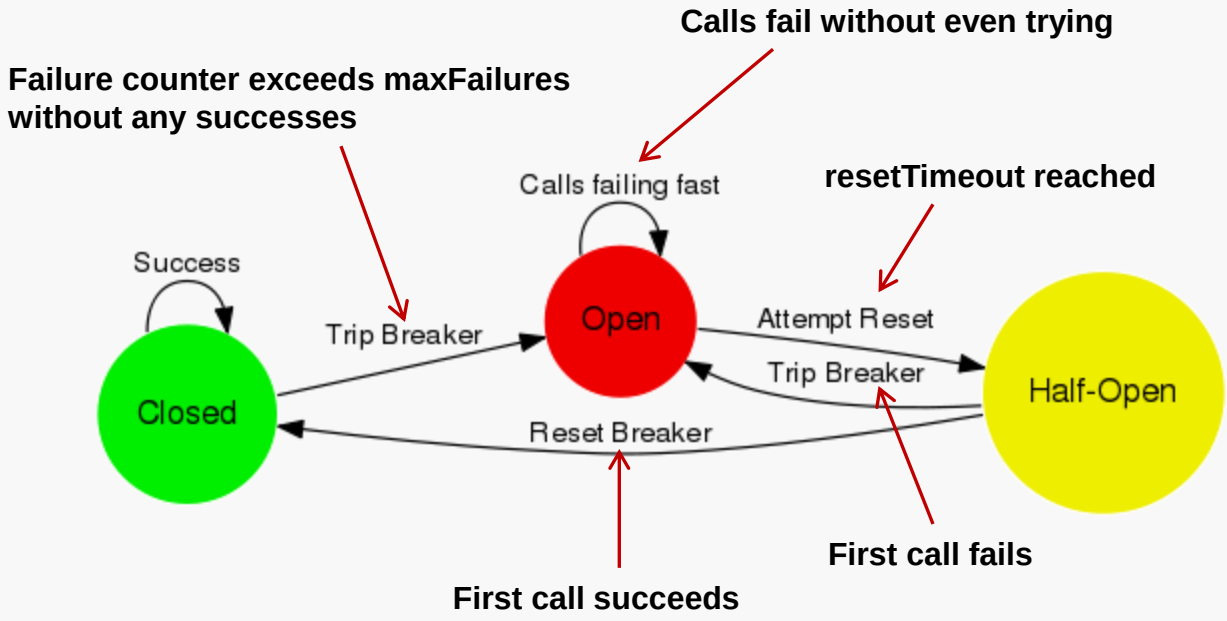
Framework allows you to replace child and continue processing pending messages.

This restart will be transparent to the rest of the system except that failed message is not processed.

Restart has a prescribed sequence of operations while leaving other parts of the system intact.

- Suspend the actor (which means that it will not process normal messages until resumed), and recursively suspend all children
- Call the old instance's preRestart hook (defaults to sending termination requests to all children and calling postStop)
- Wait for all children which were requested to terminate (using context.stop()) during preRestart to actually terminate; this—like all actor operations—is non-blocking, the termination notice from the last killed child will effect the progression to the next step
- Create new actor instance by invoking the originally provided factory again
- Invoke postRestart on the new instance (which by default also calls preStart)
- Send restart request to all children which were not killed in step 3; restarted children will follow the same process recursively, from step 2
- Resume the actor

To prevent the backup of cascading failures, use a circuit breaker in the message path.



Actors can monitor the lifecycle of other actors.

- This is usually called DeathWatch.
- Monitoring is reacting to termination; supervision is reacting to failure.
- Monitoring actor will receive a Terminated message with a default behavior to throw DeathPactException.