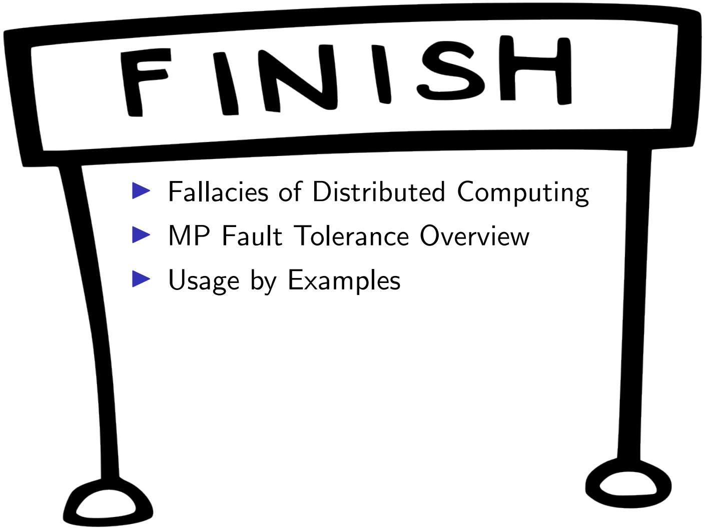
Distributed Applications may fail

Be prepared with MicroProfile Fault Tolerance

Bernd Müller
Ostfalia

Goals

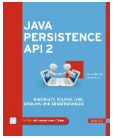


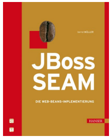
- Fallacies of Distributed Computing
- MP Fault Tolerance Overview
- Usage by Examples

Speaker

- Prof. Computer Science (Ostfalia, HS Braunschweig/Wolfenbüttel)
- Book Author (JSF, JPA, Seam, ...)

JavaServer Faces und Jakarta Server Faces 2.3
Ein Arbeitsbuch für die Praxis
HANSER





- Member EGs JSR 344 (JSF 2.2) und JSR 338 (JPA 2.1)
- Managing Director PMST GmbH
- JUG Ostfalen Co-Organizer
- bernd.mueller@ostfalia.de
- @berndmuller @berndmuller@fosstodon.org BerndMuller

Fallacies of Distributed Computing

Fallacies of Distributed Computing

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn’t change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

switches MTBF $>$ 100,000 h
power failures
human mistakes
...

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

it is not

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

constantly getting better but not infinite

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

many bad guys out there (no help from FT)
exception: DOS attacks

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

topology is changing constantly
sorry, no help from FT

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure

6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

in case whom to contact?
sorry, no help from FT

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure

7. Transport cost is zero
8. The network is homogeneous

tech: marshal/unmarshal, latency (2), ...
economically: you have to pay your cloud bill
sorry, no help from FT

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure

8. The network is homogeneous

Windows / Linux / Unix
CP-1252 / UTF-??
sorry, no help from FT

Fallacies of Distributed Computing

L. Peter Deutsch (1994), James Gosling (1997)

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure

MicroProfile Fault Tolerance at a Glance

MicroProfile Fault Tolerance at a Glance

MicroProfile Fault Tolerance at a Glance

- Implemented as CDI annotation, bound to Interceptors
- Implemented in WildFly, Open Liberty, Payara, TomEE, Quarkus, ...
- Fault Tolerance 4.0.2 included in MicroProfile 6.0
- Annotations: `@Fallback`, `@Retry`, `@CircuitBreaker`, `@Timeout`, `@Bulkhead`, `@Asynchronous` ??

The Annotations

The Annotations

The Annotations

`@Timeout`

- If guarded method takes too long, the caller will get a TimeoutException

The Annotations

`@Timeout`

- If guarded method takes too long, the caller will get a TimeoutException
- Parameters
  - `value`: the timeout value (default: 1000)
  - `unit`: the timeout unit (default: `ChronoUnit.MILLIS`)

The Annotations

@Retry

- Method should be called again in case a previous execution thrown an exception

The Annotations

@Retry

- Method should be called again in case a previous execution thrown an exception
- Parameters
  - `maxRetries`: the maximum number of retries (default: 3)
  - `delay`: delays between each retry (default: 0)
  - `delayUnit`: the delay unit (default: `ChronoUnit.MILLIS`)
  - `maxDuration`: maximum duration to perform the retry for (default: 180000, 0 means not set)
  - `durationUnit`: duration unit (default: `ChronoUnit.MILLIS`)
  - `jitter`: the random vary of retry delays (default: 200)
  - `jitterDelayUnit`: the jitter unit (default: `ChronoUnit.MILLIS`)
  - `retryOn`: exception types to retry on (default: `Exception`)
  - `abortOn`: exception types to abort on (default: {})

The Annotations

`@Fallback`

- Alternative execution path in case of failure

The Annotations

`@Fallback`

- Alternative execution path in case of failure
- Parameters
  - `applyOn`: the list of exception types which should trigger fallback
  - `fallbackMethod`: the method name to fallback to
  - `skipOn`: the list of exception types which should not trigger fallback
  - `value`: the fallback class to be used

The Annotations

`@Bulkhead`

- Limit the number of concurrent calls to prevent faults to cascade

The Annotations

`@Bulkhead`

- Limit the number of concurrent calls to prevent faults to cascade
- 2 alternatives
  - Thread pool isolation (w/ @Asynchronous, concurrent requests + waiting queue size)
  - Semaphore isolation (w/o @Asynchronous, only concurrent request)
- Parameters
  - `value`: the maximum number of concurrent calls to an instance (default 10)
  - `waitingTaskQueue`: the waiting task queue (default 10)

`@CircuitBreaker` (Michael Nygard, Martin Fowler)

- Prevent further damage by not executing functionality that is doomed to fail (fail fast)

`@CircuitBreaker` (Michael Nygard, Martin Fowler)

- Prevent further damage by not executing functionality that is doomed to fail (fail fast)
- Three circuit states: closed, open, half-open
- Parameters
  - `delay`: the delay after which an open circuit will transitions to half-open state (default 5000)
  - `delayUnit`: unit of the delay (default ChronoUnit.MILLIS)
  - `failOn`: list of exception types which should be considered failures (default Throwable)
  - `failureRatio`: ratio of failures within the rolling window that will trip the circuit to open (default 0.5)
  - `requestVolumeThreshold`: the number of consecutive requests in a rolling window (default 20)
  - `skipOn`: list of exception types which should not be considered failures (default empty list)
  - `successThreshold`: the number of successful executions, before a half-open circuit is closed again (default 1)

The Annotations

`@Asynchronous`

- Wrap the execution and invoke it asynchronously
- No parameters
- Same as in Jakarta Enterprise Beans and Jakarta Concurrency

Some Remarks for Completion

Some Remarks for Completion

Some Remarks for Completion

Remarks

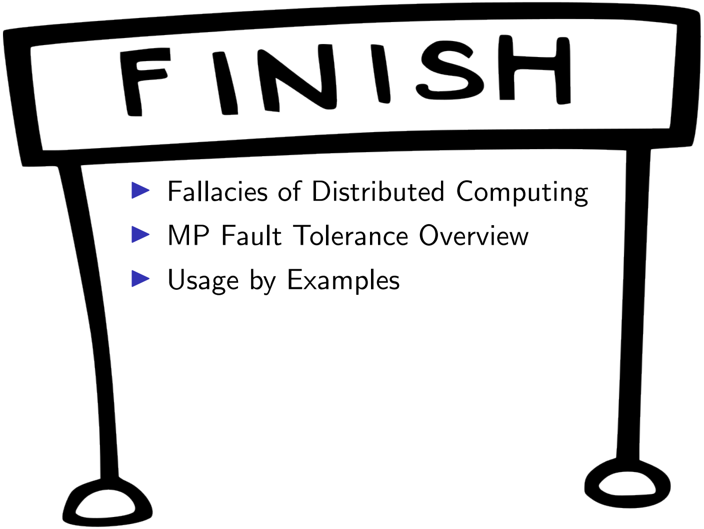
- Annotations on methods take precedence over the same annotation on class level
- If combined defined sequence: Fallback, Retry, CircuitBreaker, Timeout, Bulkhead
  i.e. if method times out, it is retried. If no attempt succeeds, fallback is used
- Priority starting at `Priority.PLATFORM_AFTER + 10` (4010)
- Annotation parameters can be configured with MP Config

Demo Time

Demo Time

Sleepy from slides, we are !

Recap



- Fallacies of Distributed Computing
- MP Fault Tolerance Overview
- Usage by Examples

Comments

Slides and Code

https://github.com/BerndMuller/fault-tolerance-eclipsecon-2023