Transactions for Microservices? Really?

Mark Little and Tom Jenkinson

October 2017

Overview

- Transactions for Microservices and why they are needed
- Eclipse MicroProfile and transactions
- Live coding demo using Narayana LRA
- Conclusions

Fault tolerance

- Machines and software fail
  - Fundamental universal law (entropy increases)
  - Things get better with each generation, but still statistically significant
- Failures of centralized systems difficult to handle
- Failures of distributed systems are much more difficult
- Microservices are all about distributed systems


3 4G
14:04
67%
Tweet
stacks machine @cemerick · 05/01/2015
Uh, microservices. So, people are hooking minute bits of computation together via unmanaged pipes carrying opaque chunks of encoded data?
9
36
44
Christian Posta Retweeted
stacks machine
@cemerick
Replying to @cemerick
Microservices, because designing, implementing, deploying, monitoring, managing, and supporting network APIs is so [illegible] easy.
05/01/2015, 20:40
112 RETWEETS 109 LIKES
Tweet your reply

Monolith by Rene Aigner

The Majestic Monolith

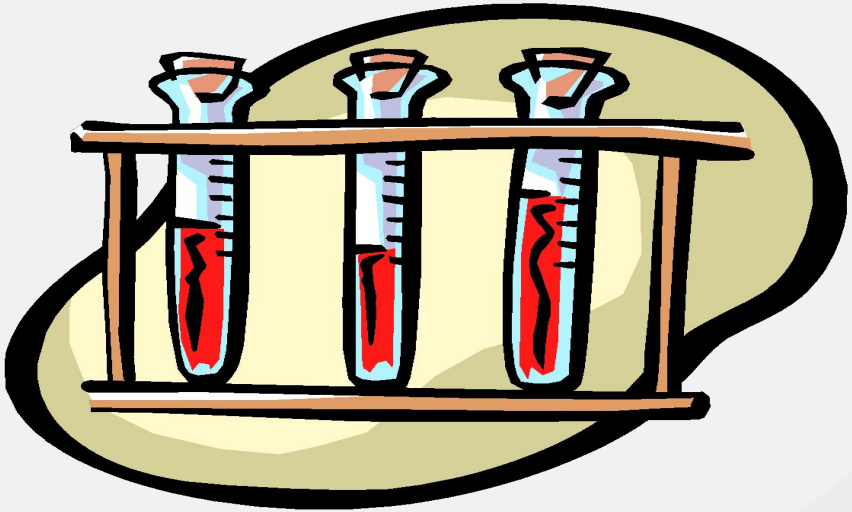

adrian cockcroft
@adrianco
Following
Replying to @kellabyte
@kellabyte @mamund I used to call what we did "fine grain SOA". So microservices is SOA with emphasis on small ephemeral components
RETWEETS
3
LIKES
5
1:16 AM - 11 Dec 2014
7
3
5

What is a transaction?

- Mechanistic aid to achieving correctness
- Provides an “all-or-nothing” property to work that is conducted within its scope
  - Even in the presence of failures
- Ensures that shared resources are protected from multiple users
- “Guarantees” the notion of shared global consensus

Properties of ACID transaction model

- Atomicity
- Consistency
- Isolation
- Durability



Transactions for MSA?

- Distributed systems interactions may be complex
  - involving many parties
  - spanning many different organisations
  - potentially lasting for hours or days
- May not be able to lock resources on behalf of an individual indefinitely
- May need to undo only a subset of work
- But ACID transactions work, right?

How about an ACID-ic MSA?

- For example:
  - JTA and XA
- Well....
  - ACID transactions implicitly assume
    - Closely coupled environment
      - All entities involved in a transaction span a LAN, for example.
    - Short-duration activities
      - Must be able to cope with resources being locked for periods
  - Therefore, do not work well in
    - Loosely coupled environments!
    - Long duration activities!

Relaxing isolation

- Internal isolation or resources should be a decision for the service provider
  - E.g., commit early and define compensation activities
  - However, it does impact applications
    - Some users may want to know a priori what isolation policies are used
- Undo can be whatever is required
  - Before and after image
  - Entirely new business processes

Relaxing atomicity

- Sometimes it may be desirable to cancel some work without affecting the remainder
  - E.g., prefer to get airline seat now even without travel insurance
- Similar to nested transactions
  - Work performed within scope of a nested transaction is provisional
  - Failure does not affect enclosing transaction

Relaxation of consistency

- ACID transactions (with two-phase commit) are all about strong global consistency
  - All participants remain in lock-step
  - Same view of transaction outcome (atomic)
- But that does not scale
  - Replication researchers have known about this for years
    - Weak consistency replication protocols developed for large scale (number of replicas and physical deployment)
  - Weak/relaxed consistency is now a generally accepted approach

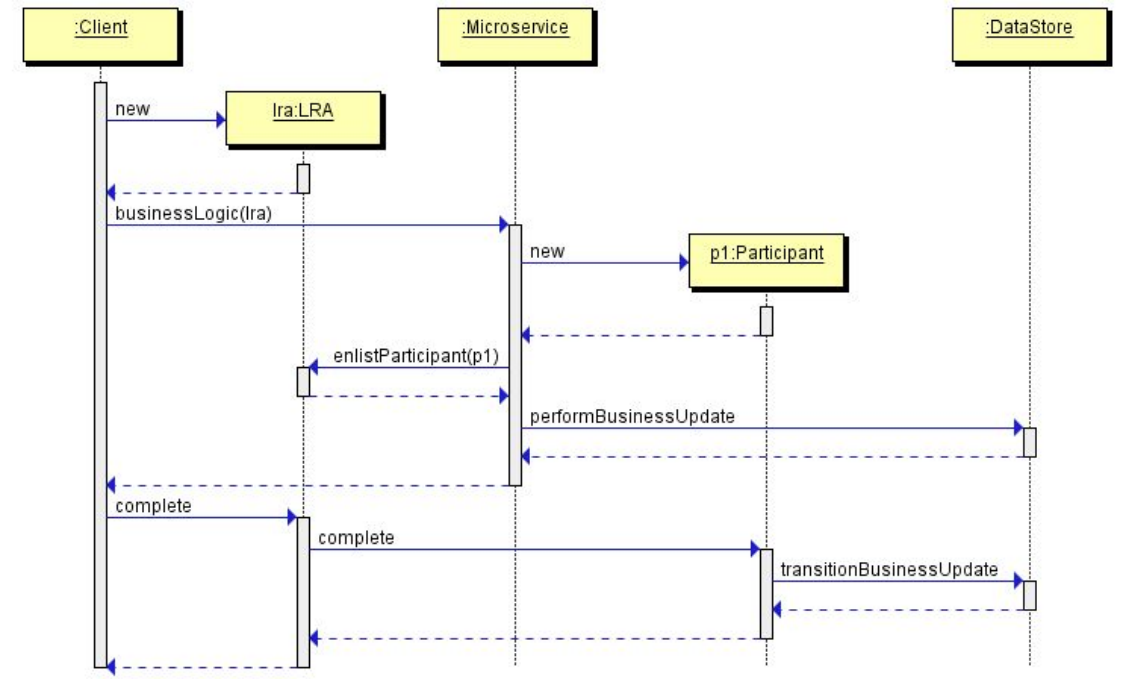
A solution: MicroProfile Long Running Action

- Currently in draft
- All parties reside within business domains
  - Business process is split into business tasks
    - Executed within own domain
  - No required notion of consistency between domains
    - Application dependent
- May need a different transaction model
  - ACID may not scale sufficiently under these conditions
- Compensatable units of work
  - Forward compensation during activity is allowed
    - Keep business process making forward progress

Highlights of the Long Running Action Spec

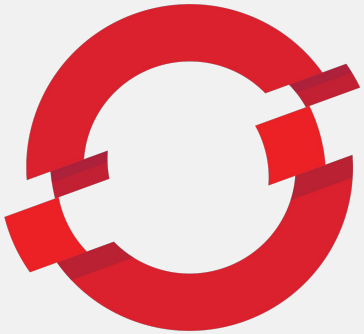
- An API for loosely coupled microservices to coordinate long running activities
- Defines:
  - REST resources to allow services written in other languages
  - CDI annotations which can be applied to JAX-RS resources
  - A pure java based coordination API

Long Running Actions: Actors and interactions



Live coding demo

Notable tech used in the demo



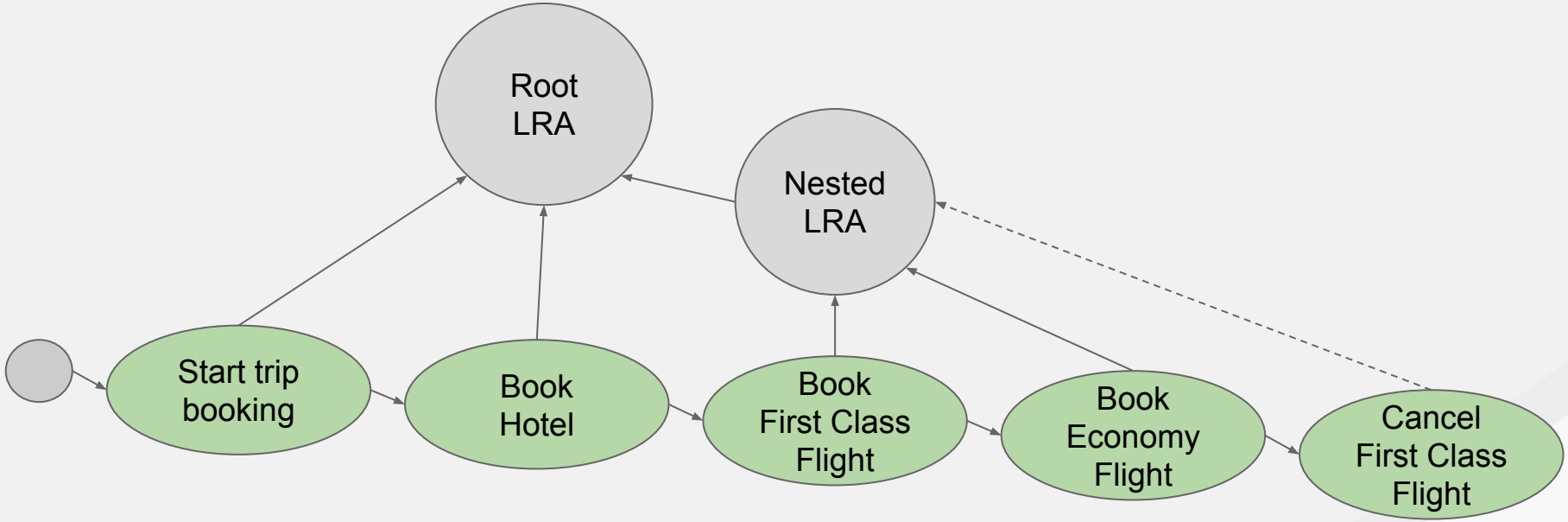
OPENSHIFT

WildFly SWARM

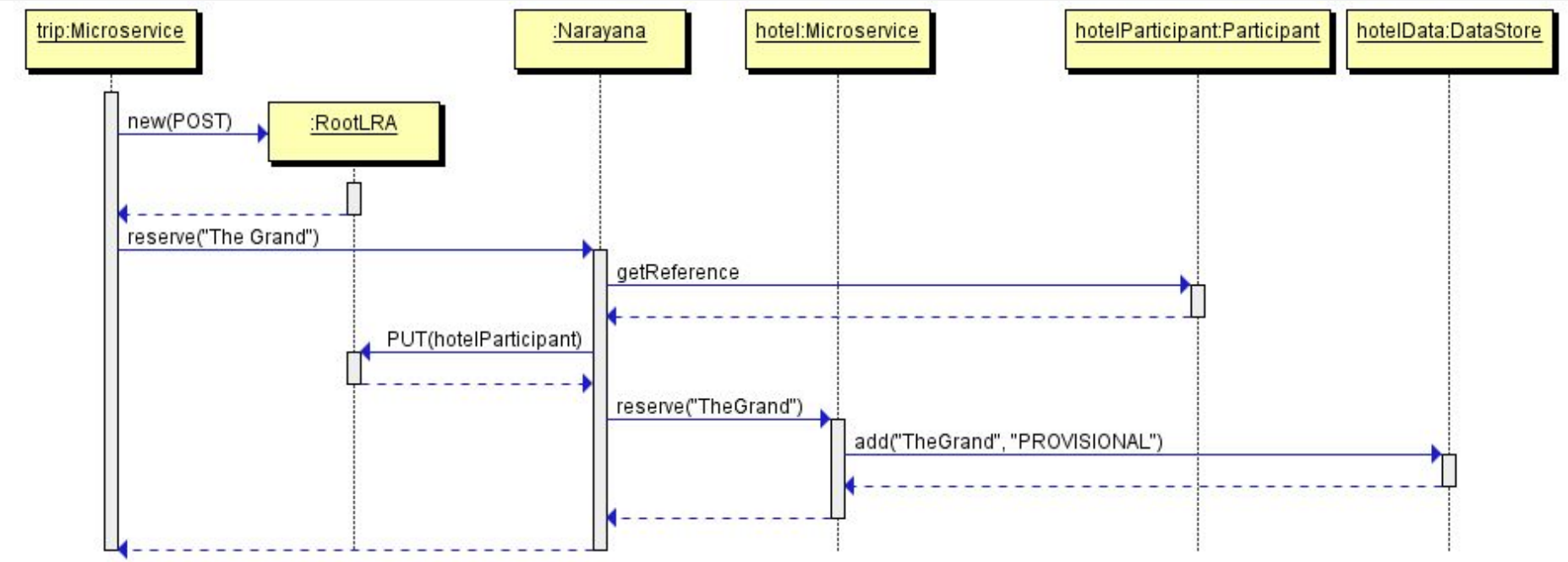
narayana

Architecture of the Demo

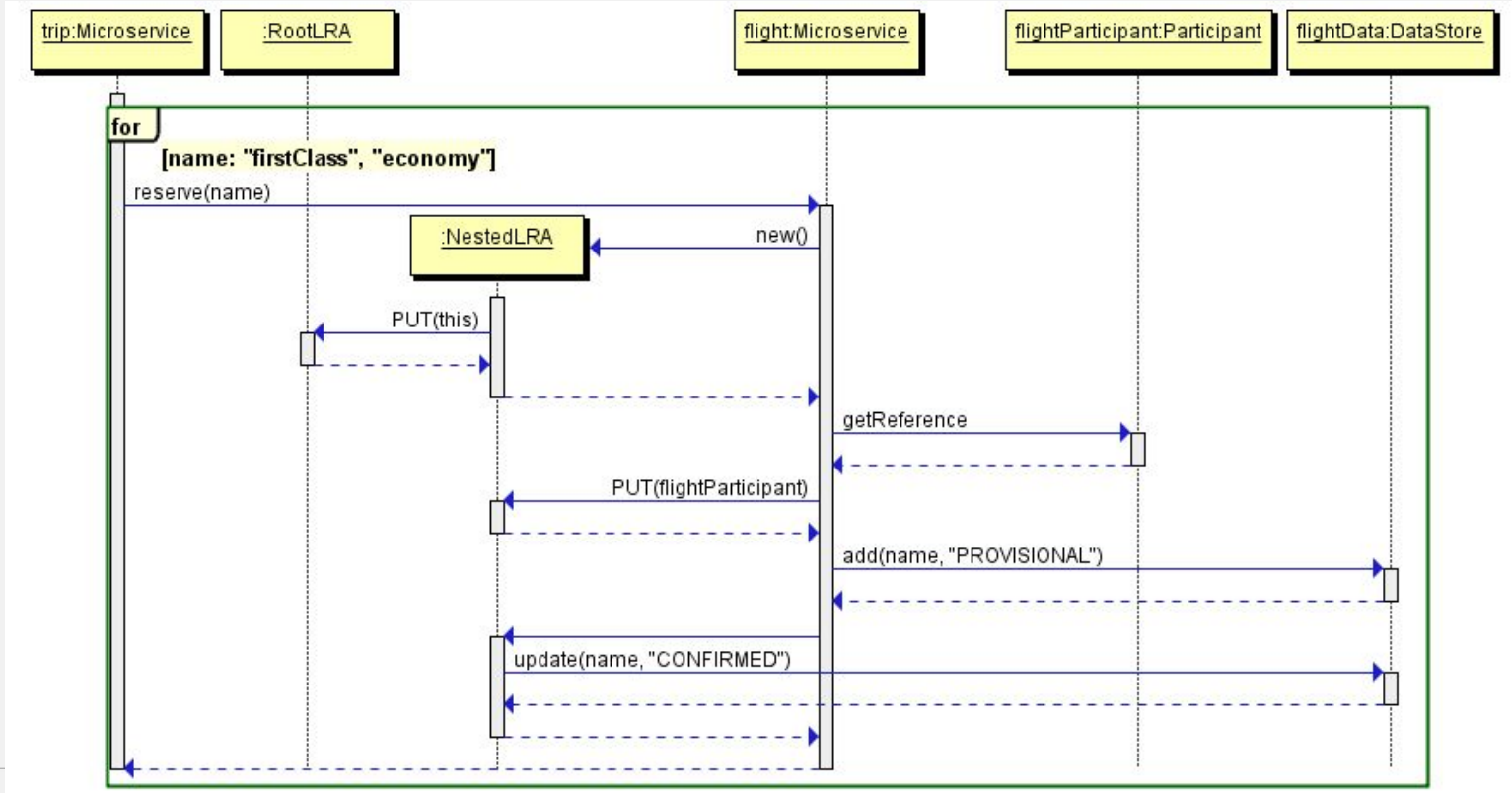
- https://github.com/jbosstm/javaone2017



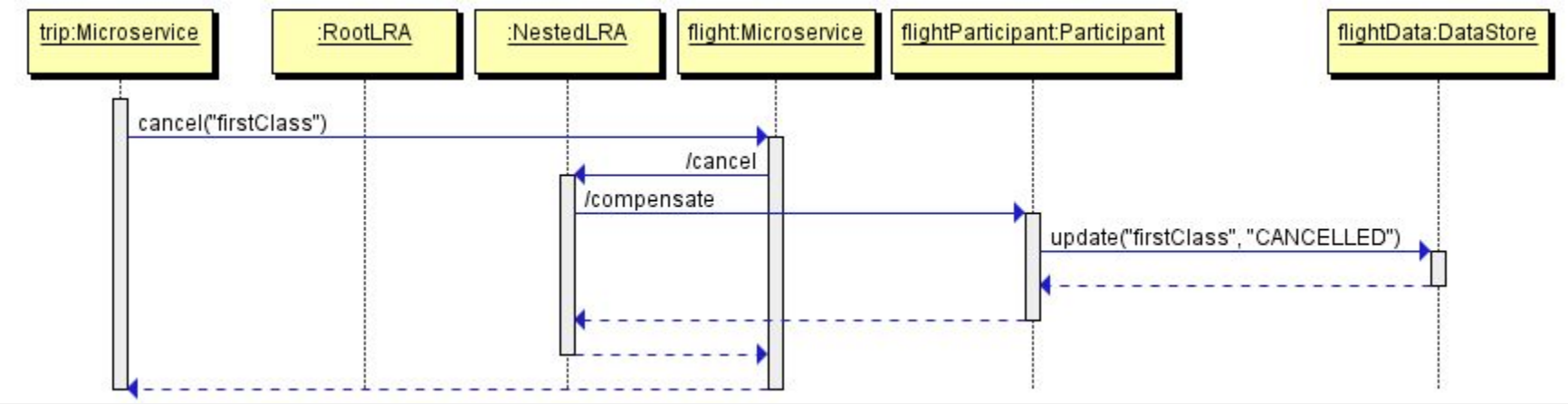
Trip Demo: Enlisting the Hotel Microservice



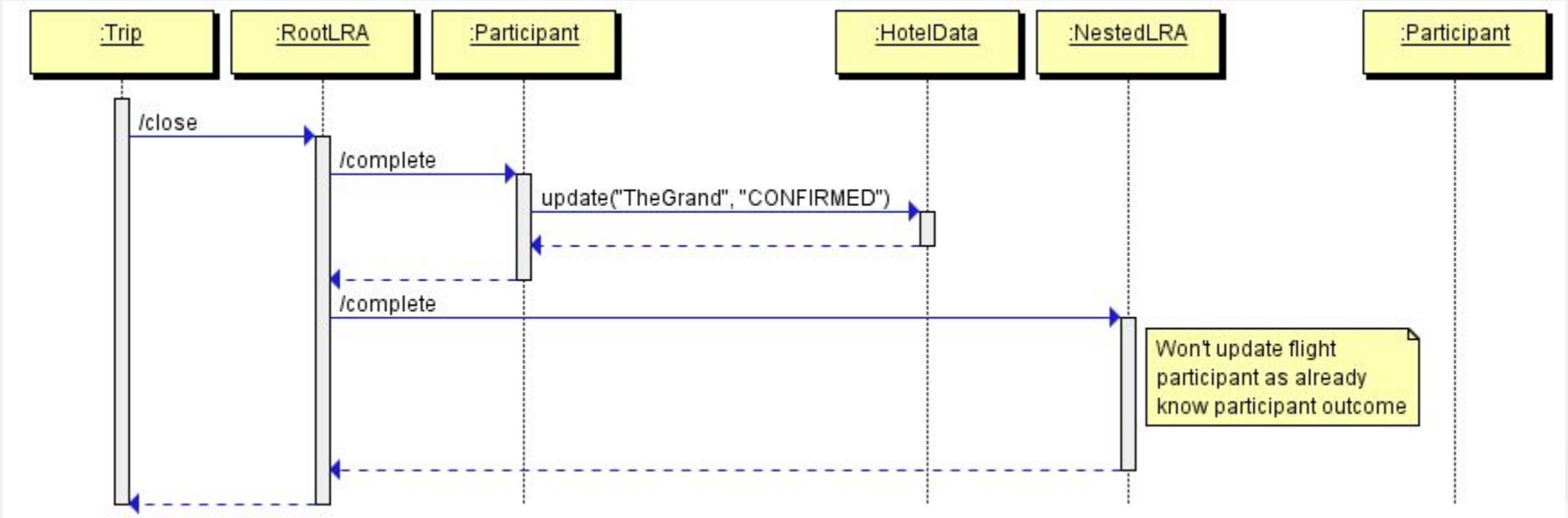
Trip Demo: Nested enlist of Flight Microservice



Trip Demo: Cancelling Nested Flight



Trip demo: Completing the booking



Where can I find out more?

Narayana.io

- Forums
- Blog
- IRC
- “The need for transactions in the Microprofile”
  - https://groups.google.com/forum/#!starred/microprofile/CJirjFkM9Do

Conclusions

- Transactions for Microservices?
  - Yes!
  - There are many transaction models that don’t necessitate full ACID
    - Long Running Action specification
- Future work in MicroProfile
  - If you don’t agree with this model then get involved and help define more
- EE4J makes life more interesting
  - Future extensions to JTA?

THANK YOU


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHatNews


youtube.com/user/RedHatVideos