Narayana: Java library for transaction processing

DevConf CZ 2021

Ondra Chaloupka

Red Hat

Goal: Overview of Narayana capabilities

Schedule: Presenting each module in few words

Red Hat

Transaction
=
unit of work

Transaction library/manager = makes model working

Transaction “model”
=
provides guarantees

- Java library and framework for transaction processing
  - https://narayana.io
  - https://github.com/jbosstm/narayana
  - https://groups.google.com/forum/#!forum/narayana-users
- Integrated in various projects
  - WildFly application runtime
  - Quarkus application framework
  - Apache Tomcat server
  - Apache Camel
  - Spring framework

Transaction Manager

Thief task

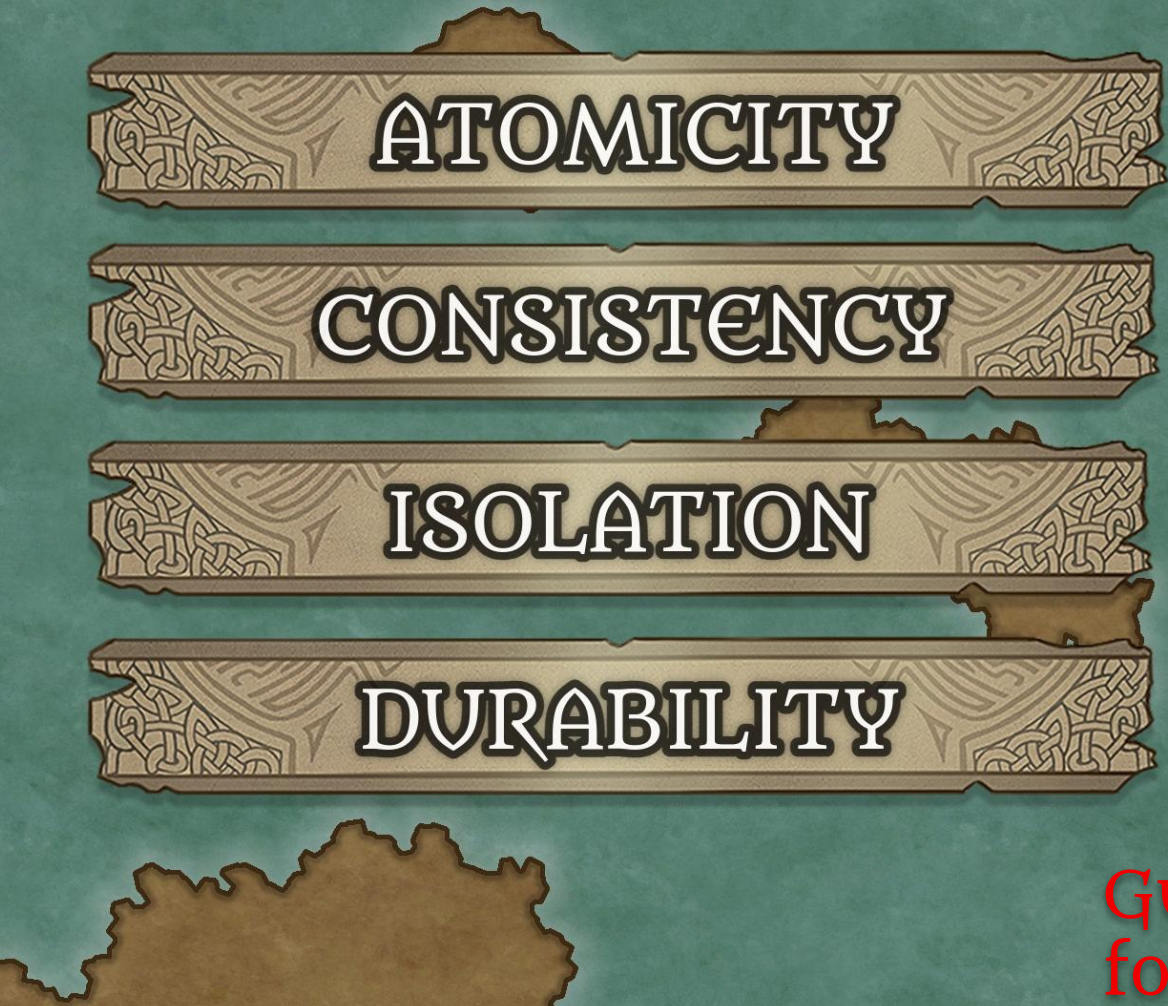
Warrior task

Cloak
resource

Picklocking
resource

Warhammer
resource


Unit of work



Guarantees
for transactional work


JTS transactions
JTA transactions
Software Transaction Memory
XTS
Long Running Action
REST - AT
World of Narayana
Transaction library


JTS transactions
JTA transactions
Software Transaction Memory
XTS
Long Running Action
REST - AT

- OTS (Object Transaction Service) standard
  - Part of the ecosystem of the ORB services
- OTS defines operations as methods on objects
  - Using IIOP as the communication protocol
- JTS (Java Transaction Service)
  - Specification for transactional interoperability between EJB containers based on CORBA, OTS and JTA

OTS Naming service

Transaction Manager

Cloak resource

resource

Thief service

Warrior service

OTD idl

```
module cz {
    module devconf2021 {
                /*
Implicit transaction propogation characteristics
                 */
                interface Thief:
CosTransactions::TransactionalObject
                {
                    /*
                     Under transactional control.
                     */
                    void help_me_warrior();
                };
    };
};
```

OTD idl

```
module cz {
    module devconf2021 {
                /*
Implicit transaction propogation characteristics
                 */
                interface Thief:
CosTransactions::TransactionalObject
                {
                    /*
                     Under transactional control.
                     */
                    void help_me_warrior();
                };
    };
};
```

protobuf idl

```
service Foo {
       rpc Bar(FooRequest) returns(FooResponse);
}
```

- ORB transactional framework to build on top of
- IIOP messages that services communicate with each other
- WildFly integrates JTS with EJB 2
  - Integrates with JTA
- “Ancient” with respect to current software development


JTS transactions
JTA transactions
XTS
Software Transaction Memory
Long Running Action
REST - AT

- JTA (Java Transaction API)
  - distributed transactions
  - X/Open XA standard uses XA resources
  - on behalf of two-phase commit protocol
- Well-known and heavily used
- Integrated with EJB and CDI in application runtimes
  - Mostly transparent to developers

Application Runtime

Transaction Manager

Warrior component

Thief component

Cloak database

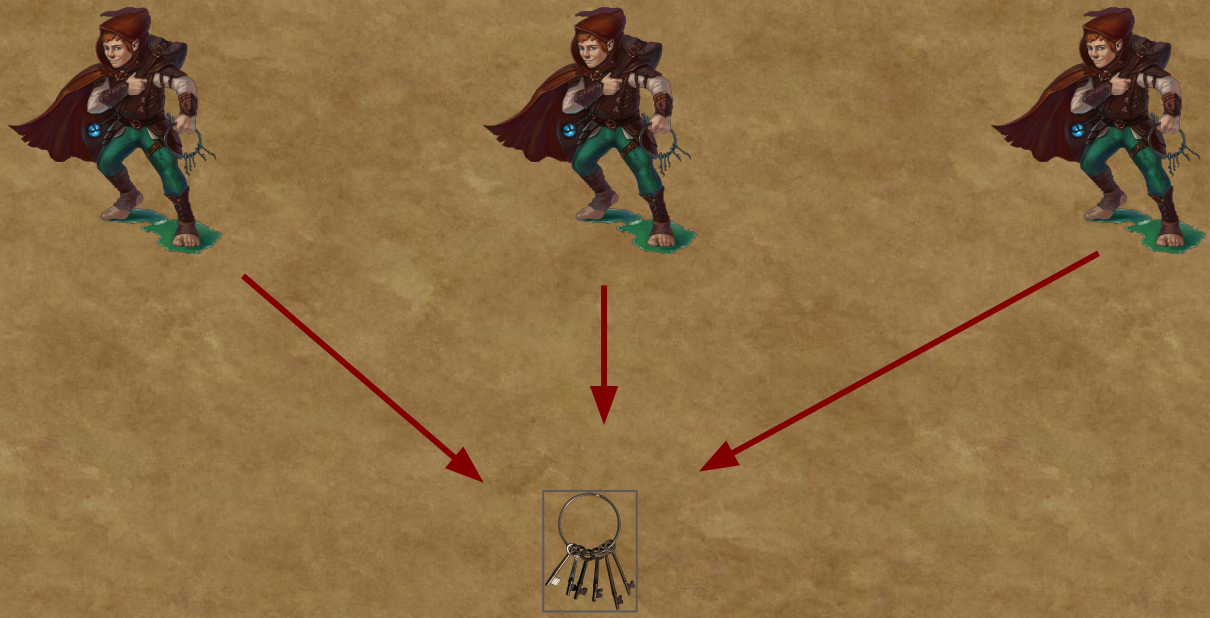
Lockpicking 3rd party resource

- Java API to work with XA transactions / 2-phase commit protocol
- Heavily used
- Integrated to various runtime
- Transparent for developer


JTS transactions
JTA transactions
Software Transaction Memory
XTS
Long Running Action
REST - AT

- STM (Software Transactional Memory)

- A concurrency models which uses shared memory
- An alternative to the lock-based synchronization approach
- Grouping memory operations to run them atomically


Lockpinging resource

```
int x = 0, y = 0, z = 0;
```

```
void first {
   synchronized(this) {
       x = x + z;
   }
}
```

```
void second {
   synchronized(this) {
       y = y + z;
   }
}
```

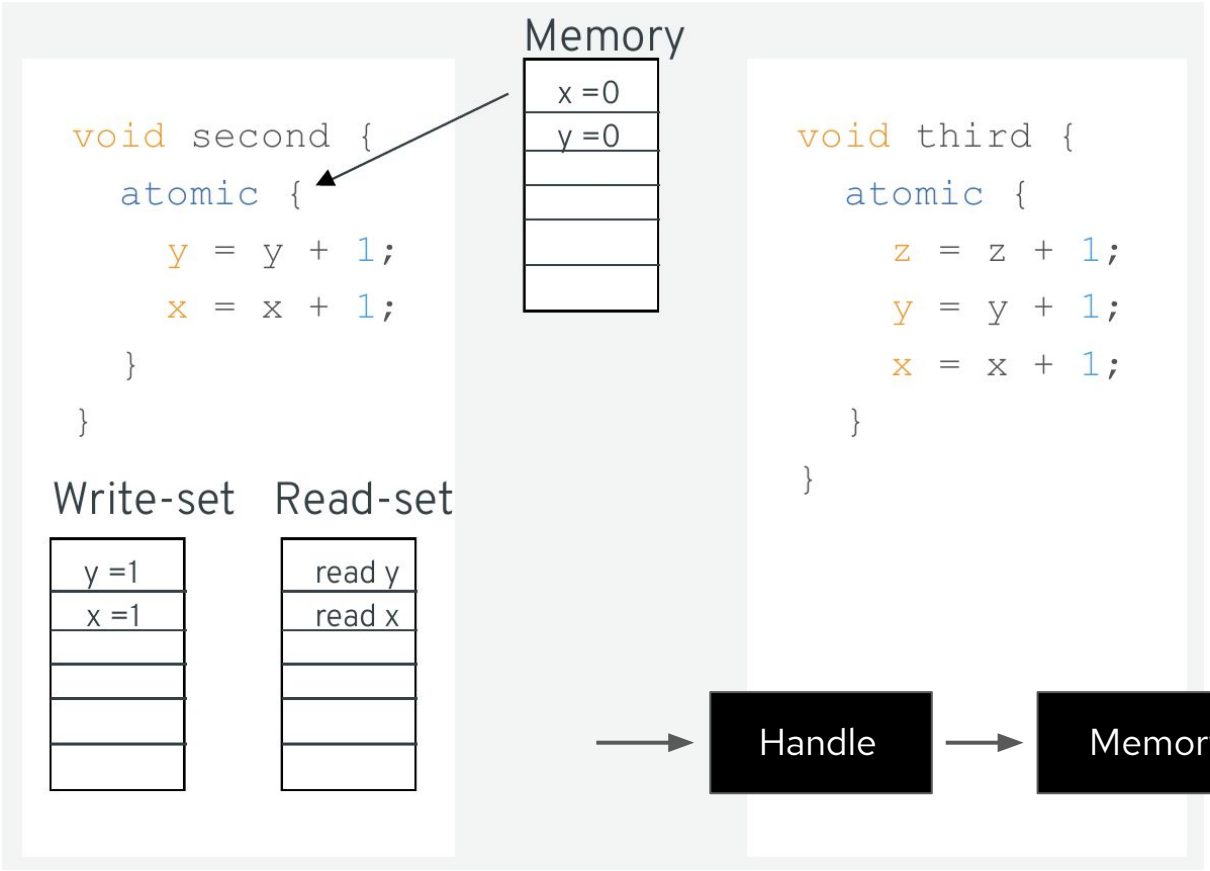
```
void third {
   synchronized(this) {
       x = x + 1;
       y = y + 1;
       z = z + 1;
   }
}
```

```
int x = 0, y = 0, z = 0;
```

```
void first {
   atomic (this) {
       x = x + z;
   }
}
```

```
void second {
   atomic(this) {
       y = y + z;
   }
}
```

```
void third {
   atomic (this) {
       x = x + 1;
       y = y + 1;
       z = z + 1;
   }
}
```


Memory
x =0
y =0
void second {
atomic {
y = y + 1;
x = x + 1;
}
}
void third {
atomic {
z = z + 1;
y = y + 1;
x = x + 1;
}
}
Write-set
Read-set
y =1
x =1
read y
read x
Handle
Memory

- optimistic locking approach for in-VM mutual exclusion

JTS transactions

JTA transactions

Long Running Action

XTS

Software Transaction Memory

REST - AT

- LRA (Long Running Actions)
  - MicroProfile specification proposal
    https://github.com/eclipse/microprofile-lra
- REST services
  - JAX-RS integration
  - Communication over HTTP
- Saga pattern
  - Not ACID - only ACD (atomicity, consistency, durability)
  - Part of responsibility moved to the service
  - Complete/Compensate callbacks

Transaction Manager
/ LRA Coordinator
(REST API)

Cloak database

Thief service

Warrior service

Warhammer database

- Annotations to declare the processing
- Saga pattern
- Business logic takes responsibility for compensation
- Services communicates with HTTP (JAX-RS)

- Session A different flavor of the distributed transaction
  - Saturday, February 20, 9:45 am


JTS transactions
JTA transactions
XTS
Software Transaction Memory
Long Running Action
REST - AT

- XTS (XML Transaction Service)
  - Transaction specification for JAX-WS (SOAP based web services)
- WS-AT (WS Atomic Transaction)
  - Atomic transactions with ACID guarantees
  - Capability of bridging JTA transactions to WS-AT
- WS-BA (WS Business Activity)
  - Saga based processing

- With WS-AT the JTA transaction may be spanned over multiple services

Cloak database

Transaction Manager

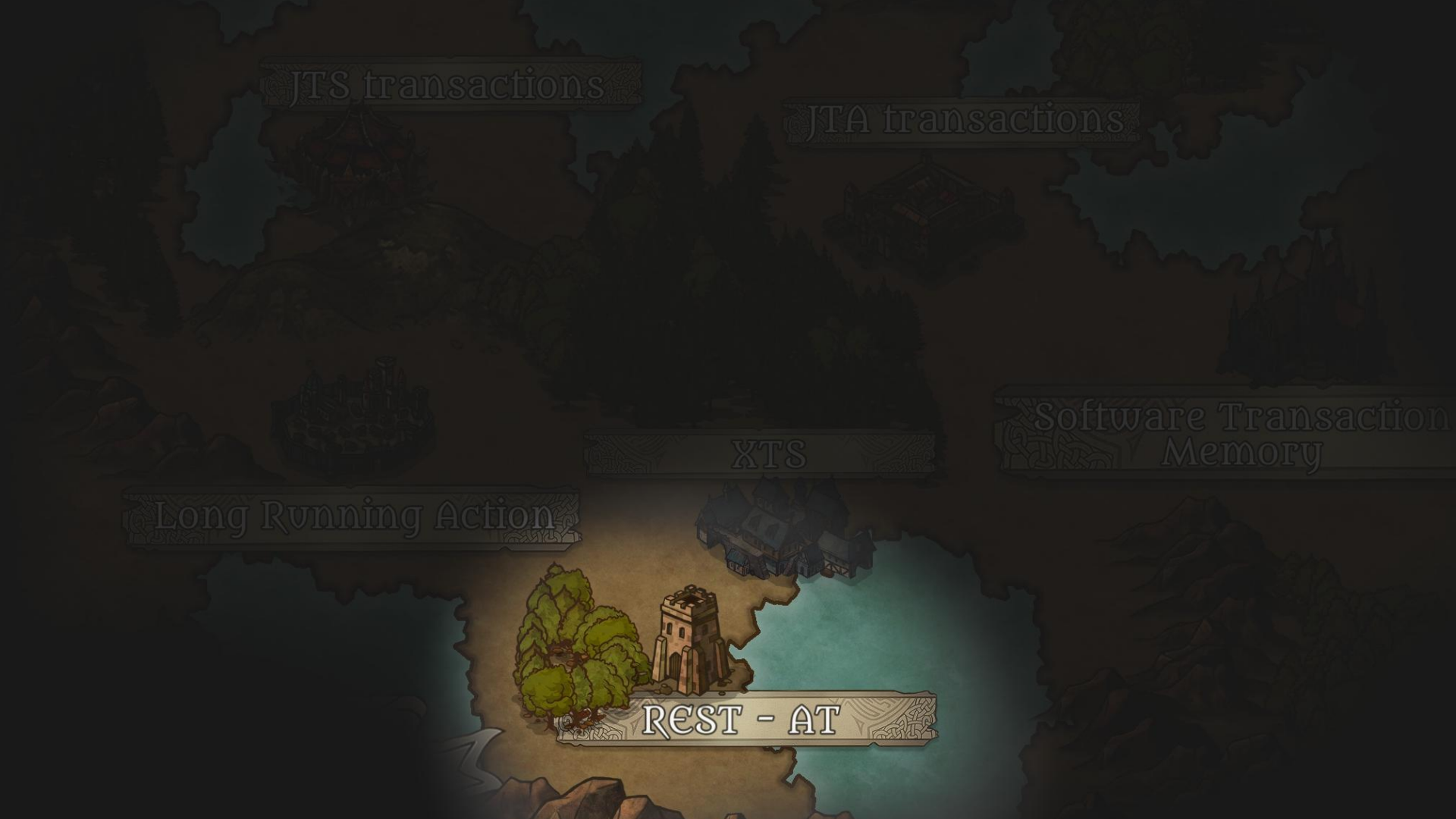
Thief component

Transaction Manager

Warrior component

Warhammer database

- Transaction over JAX-WS - SOAP based web services
- WS-AT for fully ACID / JTA transactions
- WS-BA for saga based approach


JTS transactions
JTA transactions
Software Transaction Memory
XTS
Long Running Action
REST - AT

- REST-AT (Rest Atomic Transactions)
  - Narayana protocol for handling transaction context over RESTful Web Services
- Spanning JTA transactions over HTTP calls

- Narayana modules
  - JTS
  - JTA
  - STM
  - LRA
  - XTS
  - RTS
- Slides and sources at
  - https://github.com/ochaloup/devconf2021-narayana-journey
  - https://github.com/jbosstm/quickstart