FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

comp uter lang uages

# Design and Implementation of the next Generation XVSM Framework

## Runtime, Protocol and API


Masterstudium:
Software Engineering & Internet Computing

Tobias Dönz

Technische Universität Wien
Institut für Computersprachen
Arbeitsbereich: Programmiersprachen und Übersetzerbau
Betreuerin: A.o. Univ. Prof. Dr. Dipl.-Ing. eva Kühn


## Context

**Tuple Space Middleware**
- Based on Linda (Gelernter, 1985)
- Shared data store (the "space")
- For loosely-coupled systems
  - Time and space decoupling
- Tuples have no order
  - Difference to message queues
- Getting tuples may block
  - When no matching tuple available
  - Difference to DB queries (SQL)

**Basic tuple space operations**
- *write(tuple)*: non-blocking
- *read(template)*: blocking variant
- *take(template)*: blocking variant
  - Consuming read
- *notify(template, listener)*:
  - Event registration

**Template matching**
- Template is like a tuple, but:
  - Fields are values or wildcards

## Motivation and Requirements

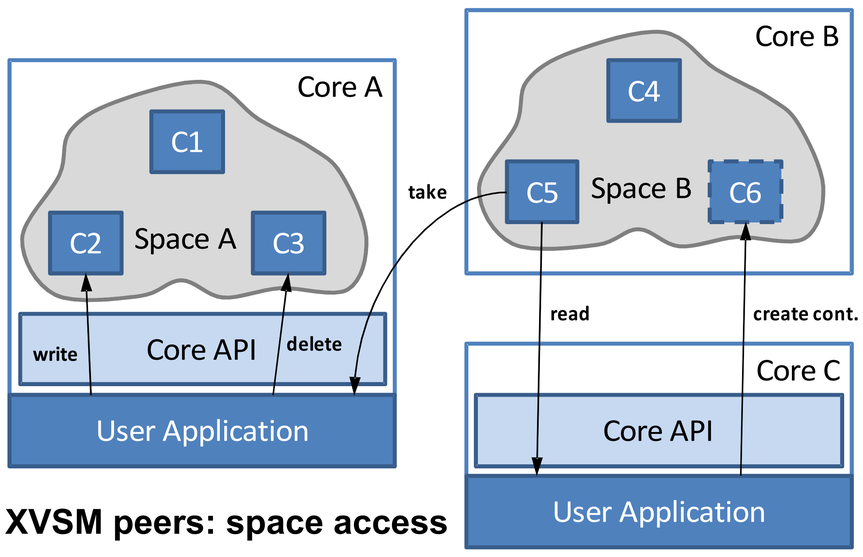
Formal model of XVSM recently specified (Craß, 2010)
- Layered architecture (CAPI layers), Haskell prototype
- Lower layers (up to CAPI-3) implemented in Java (Barisits, 2010)

MozartSpaces 2.0: Runtime, Protocol, API
- Built upon CAPI-3 impl. (containers, coordinators, transactions)
- Conformance to formal model
- Interoperable XML protocol (XVSMP)
- Synchronous and asynchronous API
- Better performance and scalability than version 1.0

## eXtensible Virtual Shared Memory (XVSM)

**Basic Concepts**
- Containers to structure space
  - Can have name and max. size
- Coordinators manage entries (tuples)
  - Flexible, not only templates
  - Can be user-defined
  - Predefined: FIFO, Key, Label, Vector, Linda, XVSM Query, ...
- Transactions (TX): with timeout
  - Multiple isolation levels possible
- Aspects (Interceptors) for extensions

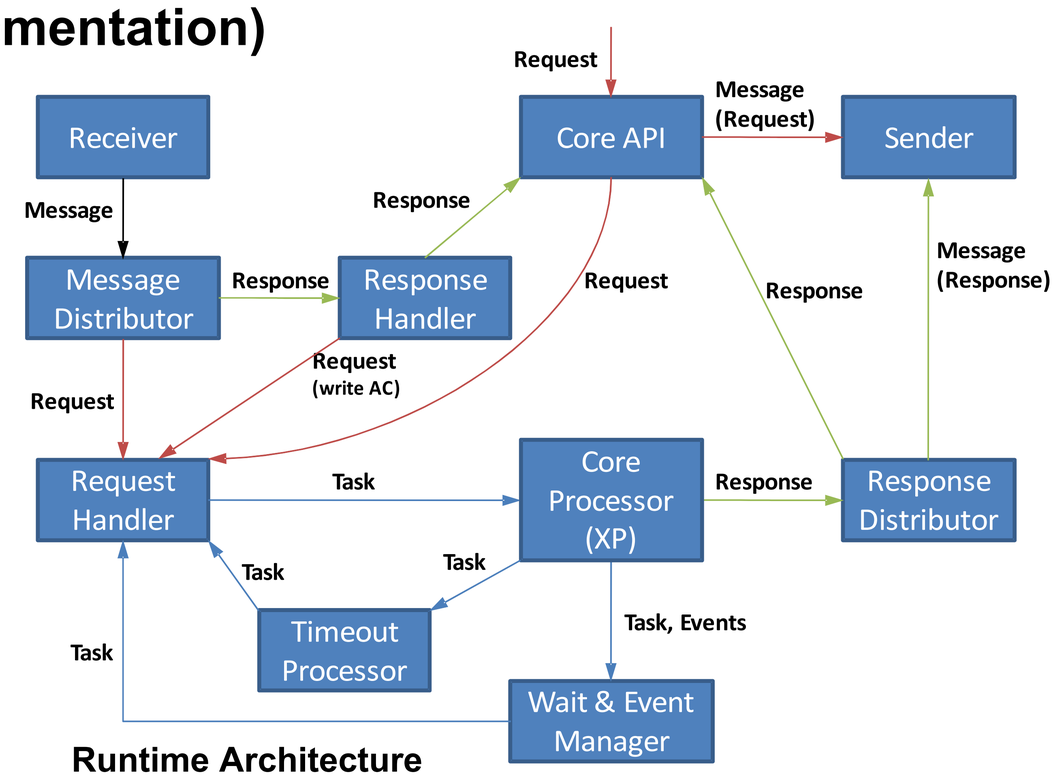
**XVSM Operations**
- Container: *create*, *destroy*, *lock*
  - *lookup* (by name)
  - Have coordinator(s)
- Entries: *write*, *read*, *take*, *delete*
  - *write* can also block (cont. full)
  - Selector(s) to get entries
- TX: *create*, *commit*, *rollback*
- Aspects: *add*, *remove*
  - Before/after operations




**XVSM peers: space access**

## MozartSpaces (Implementation)

**Runtime**
- Blocking operations with timeouts
- Requests from embedded API or remote
- Processes requests (read, write, ...) in XP
  - A task for each request
  - Tasks can be blocked (wait container)
  - Internal events to wake up tasks
  - Deadlock detection for TX
- XP: thread pool for request tasks
  - Invokes aspects before/after operation
  - Calls operation (e.g., CAPI-3)
    - They are non-blocking
- Wait & event management logic
  - Selective: considers container, TX
  - Prevents race-conditions




**Runtime Architecture**

**Protocol, Remote Access**
- XVSMP based on XML schema
- Several serializers (XVSMP, binary)
- TCP socket connections

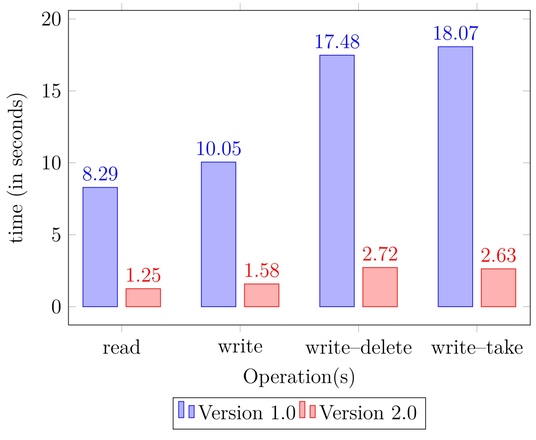
**API**
- Synchronous or asynchronous
  - with poll object or callback
- Optional answer container (in space)
  - Result stored there

**Event Notifications**
- Implemented with aspects
- Use basic XVSM operations
- Have own API
- Example of XVSM extension (profile)

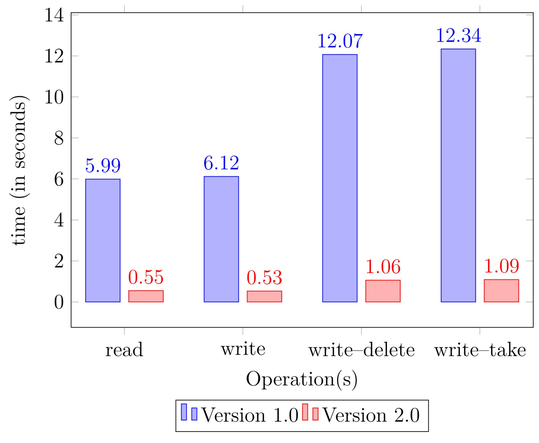
## Results

MozartSpaces 2.0 has better performance than version 1.0:
- Considerably faster for all operations, remote and embedded
- Scales also better

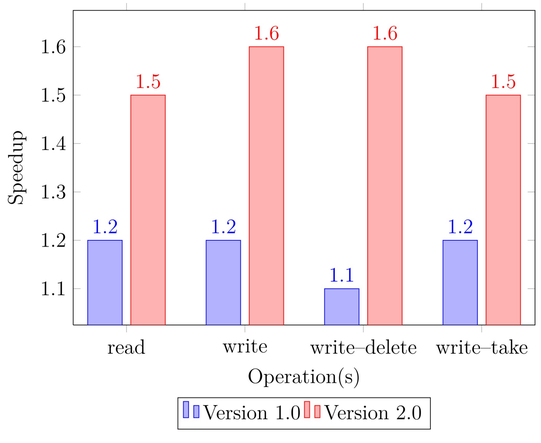
Insights from analyzing the internals:
- Timestamping (System.nanoTime) is slow
- Large messages with built-in serialization




**100 000 embedded operations**




**1000 remote operations**




**Scalability (serial vs. concurrent)**
Tested with 1 and 5 threads, 2 core machine

## Conclusion

**Requirements fulfilled**
- Implementation of formal model
- Better performance than version 1.0
- Basis for research projects
- Asynchronous API
- Interoperable XVSMP
- Analysis of internal processing
  - Some interesting insights

**Future work**
- Profiles for research projects
- Even better performance
- Alternative remote interfaces (REST, non-blocking socket I/O)

MozartSpaces is open source, download it at www.mozartspaces.org.


Kontakt: tobias@complang.tuwien.ac.at