FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

Lifecycle and Memory Management for eXtensible Virtual Shared Memory (XVSM)

Masterstudium:
Software Engineering and Internet Computing

Harald Watzke

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

Context

eXtensible Virtual Shared Memory (XVSM)

- Space Based Architecture (SBA)
  - Tuple Space Paradigm, shared data store (“space“)
  - Based on Linda (Gelernter, 1985)
- Middleware consists of multiple, connected XVSM spaces
- Layered architecture
- Containers structure space
- Coordinators on containers manage data (tuples)
  - FIFO, Key, Label, Vector, Linda, Query provided
- Container-Operations: *create, destroy, lock, lookup*
- Entry-Operations: *write, read, take (consuming read), delete*
  - Selectors used to select entries using coordination data from coordinators
- Aspects provide extensibility and programmability (AOP)
- Transactions (with timeouts)

Motivation and Requirements

MozartSpaces

- Java Reference Implementation, based on formal model
- Existing event architecture limited to task handling, not extensible
- No support for time-based or custom events
- Aspects wrap around operations, no operation-independent alternative extensibility mechanism
- No information lifecycle management, specifically disposal phase

Requirements

- Event-driven extensibility mechanism, comparable to Aspects
- Integrated information lifecycle management
- Support alternative handling of expired entries
- Limit impact on MozartSpaces architecture, implement only on top layer (runtime)
- Low performance overhead

Extended MozartSpaces Runtime

Event Architecture

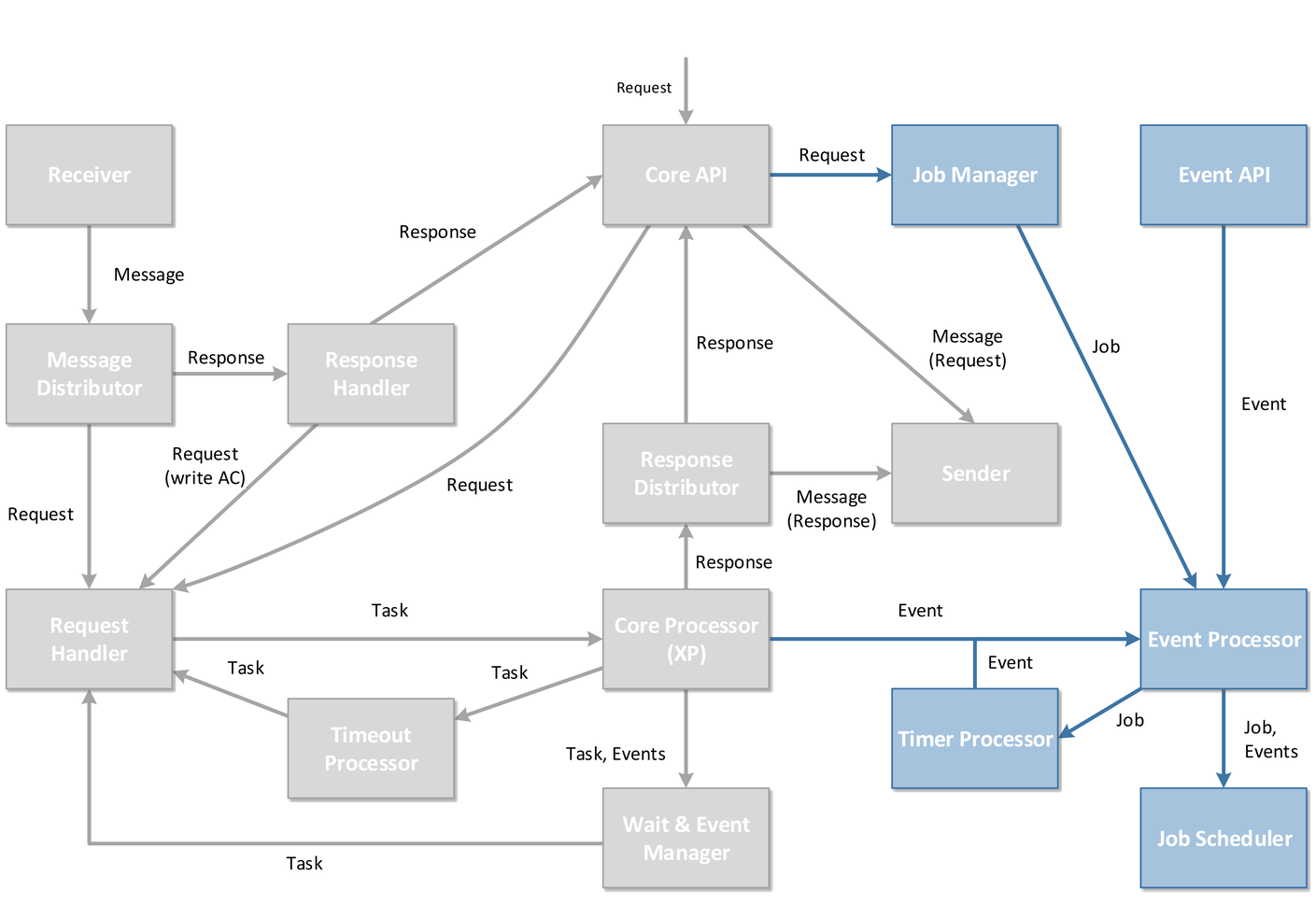
- Extensible, comprehensive event architecture
  - Events triggered by space operations (Write, Read, ...)
  - Time-based events, triggered by Timer Processor
  - Custom events supported
- Non-Blocking
- Jobs register expected event conditions
  - Complex event combinations supported
  - Templates specify expected events
  - Templates logically AND/OR related
- Components use event dispatchers to trigger events

Job Scheduling

- Jobs execute arbitrary code
  - Including operations on the space
- Installed long-term, repeated executions
- Triggered by events
- Priority Scheduling

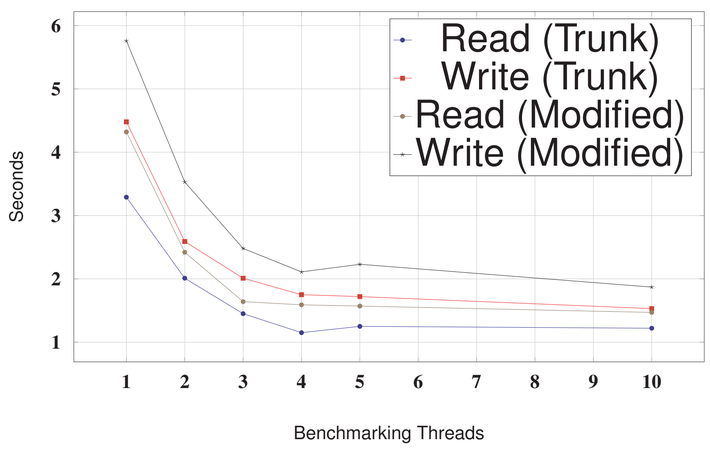
Leasing

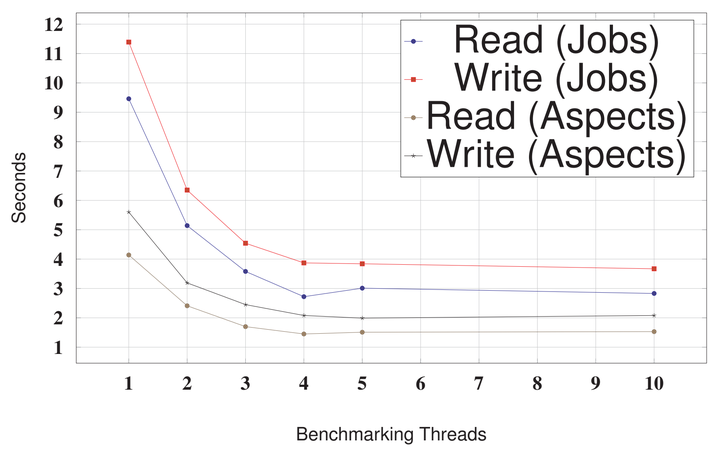
- Entries have lease duration, i.e. lifetime, validity
- Time-based reaping of expired entries
- Default: Delete expired entries
- Custom reaping behavior supported



Results

Small overhead produced by new mechanisms. Jobs perform slightly slower than aspects, however aspects execute as part of space operation tasks, jobs execute after tasks





1.000.000 write/read operations

Conclusion

Requirements fulfilled

- Extensible event architecture
- Event-driven extensibility mechanism (jobs)
- Leasing removes expired entries, frees up space
- Performance impact within expectations

Future Work

- Rework aspect and task scheduling systems to use new mechanisms
- More complex event conditions

Kontakt: harald.watzke@alumni.tuwien.ac.at