Diplomarbeitspräsentationen der Fakultät für Informatik

A Formal Model of the Extensible Virtual Shared Memory (XVSM) and its Implementation in Haskell

Design and Specification

Stefan Craß

Masterstudium:
Software Engineering & Internet Computing

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

Context

The coordination of processes in distributed applications is a difficult task. Therefore, middleware systems are used to hide some complexity from the developer. In particular, space-based computing (SBC) middleware enables efficient ad-hoc collaboration between distributed peers with loose coupling. The coordination is realized via data tuples that are written into a shared space and can be retrieved by other peers, thus allowing a data-driven interaction style.

However, current SBC middleware systems often do not provide sufficient mechanisms to realize complex collaboration patterns. Therefore, the coordination logic has to be implemented in the user application. The XVSM concept has been developed to solve this problem.

Challenges

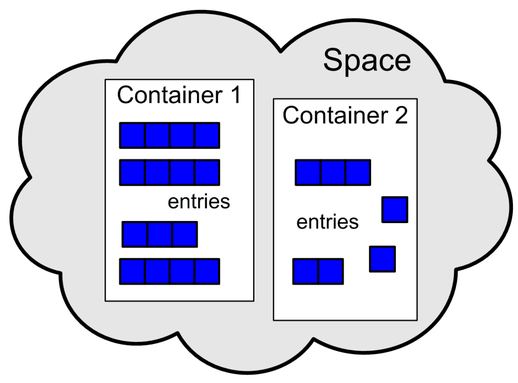
To enable the interoperability of different XVSM implementations on separate platforms, they must possess a well-defined behavior in every situation. Therefore, a formal specification has to be defined that can be used as an unambiguous reference for developers and as origin for the verification of critical parts of the middleware.

Furthermore, the formalization process should yield a clean and modular middleware architecture for XVSM with enhanced coordination capabilities and hooks for extensibility.

XVSM middleware

XVSM (e**X**tensible **V**irtual **S**hared **M**emory) is a flexible, extensible and easy-to-use SBC middleware. It enables efficient coordination between distributed processes and supports platform interoperability due to the usage of an XML protocol for communication between different XVSM runtimes.

An XVSM space consists of containers holding data entries, which are managed by one or more coordinators. These coordinators determine how entries are written into and read or taken from the container, thus defining the coordination mechanism.



Features:
- Blocking operations with timeouts
- Transactions
- Write, read, take or delete entries using arbitrary coordination laws
- User-defined coordinators
- Dynamically added aspects enriching the space semantics
- Synchronous or asynchronous invocation

Results

Formal foundation

Any XVSM space can be described as a nested data structure consisting of labeled sequences, multisets or values. On this so-called xtree, simple navigation via paths as well as several algebraic operations are defined. A powerful query language allows the selection and sorting of xtrees.

```
*/published/date < 2000 | sortup(title) | cnt(1)
```

```
[title:"JavaSpaces", author:"Freeman", author:"Hupfer",
 author:"Arnold", published:[date:1999, by:"Addison-Wesley"]]
```

The complete state of a space during runtime can be expressed with an xtree containing user and meta data. The structure of this xtree is defined in the XVSM meta model.

Layered architecture

The functionality of XVSM is realized via several Core API (CAPI) modules that access the meta model with operations of lower layers.

Protocol and APIs
- Embedded API invokes local space directly
- XML protocol used to communicate with remote spaces or APIs

CAPI-4: Runtime
- Runtime machine executes incoming user requests
- Blocked requests are rescheduled after certain events
- Aspects are invoked before or after a space operation is executed

CAPI-3: Coordination
- Non-blocking container operations
- Each coordinator has own data structures for managing entries
- Pre-defined coordinators (Fifo, Key, Query, Vector,...)

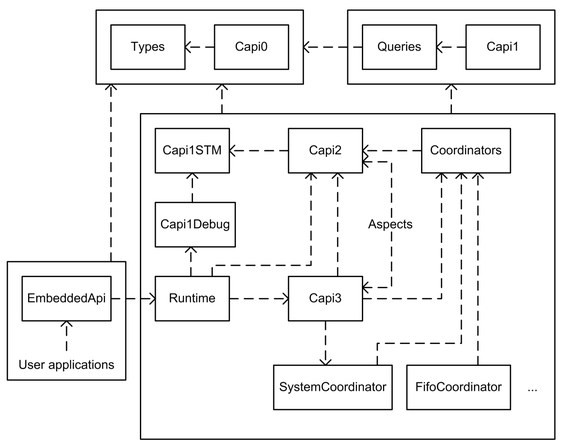
CAPI-2: Transactions
- Transaction support using pessimistic locking
- Sub transactions for internal use by runtime

CAPI-1: Basic data access
- Writing data structures to the space
- Queries on xtrees with read or take

Haskell prototype

To prove the accuracy of the XVSM specification, an executable prototype in the functional programming language Haskell has been developed. Haskell was chosen because it is easily comprehensible, provides extensive libraries and allows an implementation that corresponds closely to the formal model. In Haskell, a function's result only depends on its parameters and thus side effects like in object-oriented programming languages are prevented.

The Haskell prototype consists of several modules that correspond to the CAPI layers of a single XVSM core without remote communication. New coordinators and aspects can be added to the space with little programming effort. The successful development of the prototype shows that the specification is feasible. By examining the behavior of the prototype in various highly concurrent scenarios, possible problems can be detected, which might lead to further improvements of the formal model.



Conclusion

Complex distributed applications can be designed efficiently with space-based computing middleware. XVSM, which allows highly flexible coordination mechanisms, requires clearly specified semantics to enable the interoperability of implementations on various platforms. Based on a simple algebraic foundation and an expressive query language, the semantics of the middleware's core functionality are defined via the specification of modules for basic data access, transactions, coordination and the runtime machine. A meta model is used to bootstrap the behavior of the space with own mechanisms. It is also shown how the middleware can be adapted to support arbitrary coordination laws that exceed the default semantics. The implementation of an XVSM prototype in Haskell serves as a proof of concept for the developed model.

Based on the described specification, new implementations of XVSM will be developed that support the defined semantics. Future work also includes the modeling of the XVSM semantics using more formal approaches like process calculi or temporal logic and the definition of currently unspecified parts like the XML protocol.

Kontakt: sc@complang.tuwien.ac.at