Introduction to Middleware

Petr Tůma
DEPARTMENT OF DISTRIBUTED AND DEPENDABLE SYSTEMS
FACULTY OF MATHEMATICS AND PHYSICS
CHARLES UNIVERSITY IN PRAGUE

http://d3s.mff.cuni.cz
What is middleware anyway?

Think about programming paradigms

- functional programming
  
  “it helps to view a program as an evaluation of a complex mathematical function with no side effects”

- procedural programming
  
  “it helps to view a program as a hierarchical composition of smaller procedures”

- object oriented programming
  
  “it helps to view a program as interactions that change state of otherwise independent objects”
What is middleware anyway?

Programming paradigms give us useful concepts and tools to solve certain problems in certain situations.

Note also

- design patterns
- application frameworks
- programming methodologies
- ...
What is middleware anyway?

Middleware are **tools** (and often **concepts**) that are helpful in writing applications for **distributed** or **parallel** environments.

- how to install (deploy) applications
- how to achieve high performance
- how to communicate easily
- how to synchronize easily
- how to access data
- ...
Middleware is the stuff that system designers believe belongs in applications and application designers believe belongs in systems.

[Paraphrasing K. J. Klingenstein]
Communication
How to communicate over network?

Network sends messages, right? ... View communication as sending messages!

- Just what could the interface look like?
  send (target, message)
  receive (buffer)
- Dull?
Count all prime numbers in given range.

Trivial implementation

- Iterate over all numbers in range
- Test each number separately

Note

- Easy to split into smaller tasks
- Similar to many useful applications
Example: MPI

Standard for communication in clusters ...

- Support for complex data types
- Coordinated communication patterns
- Portability between platforms and languages

Example uses just two functions from dozens

- Scatter (in_buffer, out_buffer)
  Splits input buffer to multiple participants
- Reduce (in_buffer, operation)
  Performs operation between buffers
Implement cooperative drawing board.

Trivial implementation
- Send all drawing operations to all members
- Just draw operations as they come

Note
- Membership for drawing
- Ordering becomes important
Example: JGroups

Toolkit for multicast communication ...

- Easy use of multicast
- Membership, reliability, ordering, atomicity

Example uses these functions

- Channels with membership management
- Reliable send and receive
- Callback interface
- Serialization
Common messaging features ...

• Message structure
  ▪ Complex messages (data structures, XML)
  ▪ Message portability (automated conversion)

• Communication patterns
  ▪ Many to many communication
  ▪ Queueing, threads, callbacks

• Communication properties
  ▪ Reliability, resiliency (including persistence)
  ▪ Ordering (sender, causal, total)
Can we communicate otherwise?

Do we send messages in normal code?

... Events really delivered as **method calls**!

- Proxies on client look like real objects
- Proxy methods are stubs that send messages
- All processing really happens on the server
- No need for writing communication code
Example: RMI

Support for communication in Java ...

- Seamless integration
- Simple features

Example points to think about

- How fast can all this be?
- How are arguments passed?
- Where do threads come from?
Example: CORBA

Standard in heterogeneous environments ...

- Supports C, C++, Ada, Java, Python ...
- Many extensions
  - Real time environments
  - Embedded environments
  - Component deployment

Example points to think about
- Interface description and language mapping
- How long would this take to write manually?
Common remote call features ... 

Interface definitions
- Functions, objects, components
- Mappings to multiple languages and environments

Parallelism support
- Threading models
- Event loop integration

Protocol support
- Standard protocols (SOAP over HTTP, IIOP)
- Optimizations (shared memory, compression)
Can we communicate otherwise?

Threads on one node share memory?

... We can try **sharing memory** too!

- Intercepting memory access just iceberg tip
  - Help from virtual memory manager hardware
  - Even easier in interpreted environments
- Appropriate consistency model
  - Immediate synchronization has high overhead
  - Relaxed consistency models can be useful
- Special addressing modes
Implement distributed data storage.

Trivial implementation

- Distribute data to nodes using hashing
- Use replication for reliability

Note

- Efficient routing is essential
- Other issues like membership
Example: OpenChord DHT

Implementation of a distributed hash table ...

- Simple interface
  - put (key, value)
  - get (key)
- Logarithmic routing complexity
- Robust in face of failures

Example points to think about
- Benefits of peer to peer environments
- Applications (robust storage, caches, scale)
Synchronization
How to synchronize parallel activities?

We have threads and locks, so what?

... We need something **simple** and **scalable**!

- Threading interfaces are relatively cumbersome
- Blocking synchronization does not scale
- Fine grained locking leads to deadlocks
- Notoriously difficult to get right
Example: OpenMP

Standard for fine grained parallel constructs ...

- We do not really want to care about threads
- Just give hints where parallelism is possible

Example points to think about

- What can compiler decide by itself?
- What about asymmetric architectures?
- How about the operating system thread API?
Implement simple account transfers.

Trivial implementation
- Balance kept in simple array
- Transfer is just decrement and increment

Note
- Think about synchronization
- How likely are races or deadlocks?
Example: deuceSTM

Java transactional memory implementation ...

- Implemented as an agent
- Atomicity in annotations

Example points to think about

- Concurrency vs efficiency
- Code robustness
- Composability
Example: ProActive

Java distributed agent environment ...

- Computation done by independent agents
- Asynchronous communication
- Synchronization on futures

Example points to think about

- Relaxed synchronization
- Application of migration
Persistence
Relational databases seem to be quite good?

... So **map** our data to tables!

- Many objects-to-relations mapping technologies
- (Almost) seamless usage in source code
- But relational databases are not all
Example: Google BigTable

Highly scalable distributed storage ...

- Table with row keys, column keys, timestamps, values
- Relatively simple interface
  - Individual row manipulation atomic
    ```
    RowMutation rm (table, key);
    rm.Set (column, value);
    Operation op;
    Apply (&op, &rm)
    ```
  - Single row look up and multiple row scan
Example: Google BigTable

Highly scalable distributed storage ...

- Impressive performance
- Reported results on 1786 four core nodes, 500 GB table, 500 nodes as table servers
  - Random 1kB reads 120 MB/s
  - Sequential 1kB reads 1235 MB/s
  - Random 1kB writes 1000 MB/s
- Many interesting applications
  - Google Crawler, 800 TB table, 1 trillion cells
  - Google Earth, 70 TB table, 9 billion cells
  - ...
Moving To Clouds
Combine all this ...

How about having an environment that combines all these nifty technologies?

- Remote communication
- Transparent data persistence
- Simplified application deployment
- Management and monitoring tools
- ...

Department of Distributed and Dependable Systems
Combine even more ...

Throw in virtualization technology ...

- Computing resources become dynamic
- Virtualization possible at different levels
  - Entire operating system
    - Amazon Elastic Cloud
    - Microsoft Azure
  - Application server
    - Google AppEngine
Example: Google App Engine

Application server cloud for Java and Python ...

- Dynamic web
  - Servlets, JSP
  - Django
- Persistent storage
  - JDO, JPA
  - Datastore with query language
- Asynchronous tasks
- Load balancing
Example: Amazon Elastic Cloud

Virtualized system cloud ...

- **Storage**
  - System volumes
    - Boot from block store volume
    - Attach volume as a device
    - Snapshot and replication
  - Object buckets
    - Retrieve by key

- **Computing instances**
  - On demand
  - Reserved

- **Interfaces for controlling everything**
Courses ...

NSWI068 Objektové a komponentové systémy
- Some middleware technologies, components

NSWI080 Middleware
- Multiple middleware technologies

NDBI016 Transactions
- Theoretical background for correctness
- Algorithms, software transactional memory

NPRG042 Programování v paralelním prostředí
- Algorithms, some cluster middleware

NSWI035 Principy distribuovaných systémů
- Distributed algorithms