MASSIV

Online Games Middleware

Presentation Contents

- Introduction
- Features supported by the Massiv
- The Massiv Core use in the Demo
- Evaluation of the project
- The Demo

Introduction

Part 1.

The Massiv Project Team

- Project origins date back to November 2001
- Supervisor: Ing. Petr Tůma, Dr.
- Team members:
  - Štěpán Vondrák - stoupik@users.sourceforge.net
  - Marek Vondrák - markoid@users.sourceforge.net
  - Petr Tovaryš - boovie@users.sourceforge.net
  - Ondřej Pečta - octa@users.sourceforge.net
  - Marek Švantner - marekus@users.sourceforge.net
  - Martin Havlišta - hafik@users.sourceforge.net

The Project Goals

- Online multiplayer games kit
- Online games
  - Run 24 hours a day
  - Persistent
  - Interactive
  - Thousands of players

Basic Characteristics

- Distributive
  - The world can be simulated by multiple servers
- Object-oriented
  - The world is built up from objects that can migrate among servers
- Support for static data
  - Management and distribution of data that change rarely

Who The Massiv Focuses On

- **Middleware for non-commercial sphere:**
  - **Independent developers – Open Source**
  - **Portability**
    - **Win32, Linux**
  - **Can not assume that servers are deployed to a single LAN**

The Massiv Project Components

- The Core
  - Object-oriented distributed system
  - The library sources
  - Builder tool
- The Demo
  - Simple exemplary online game
  - Demonstration of the Core features
  - Tools used to configure and manage the Demo

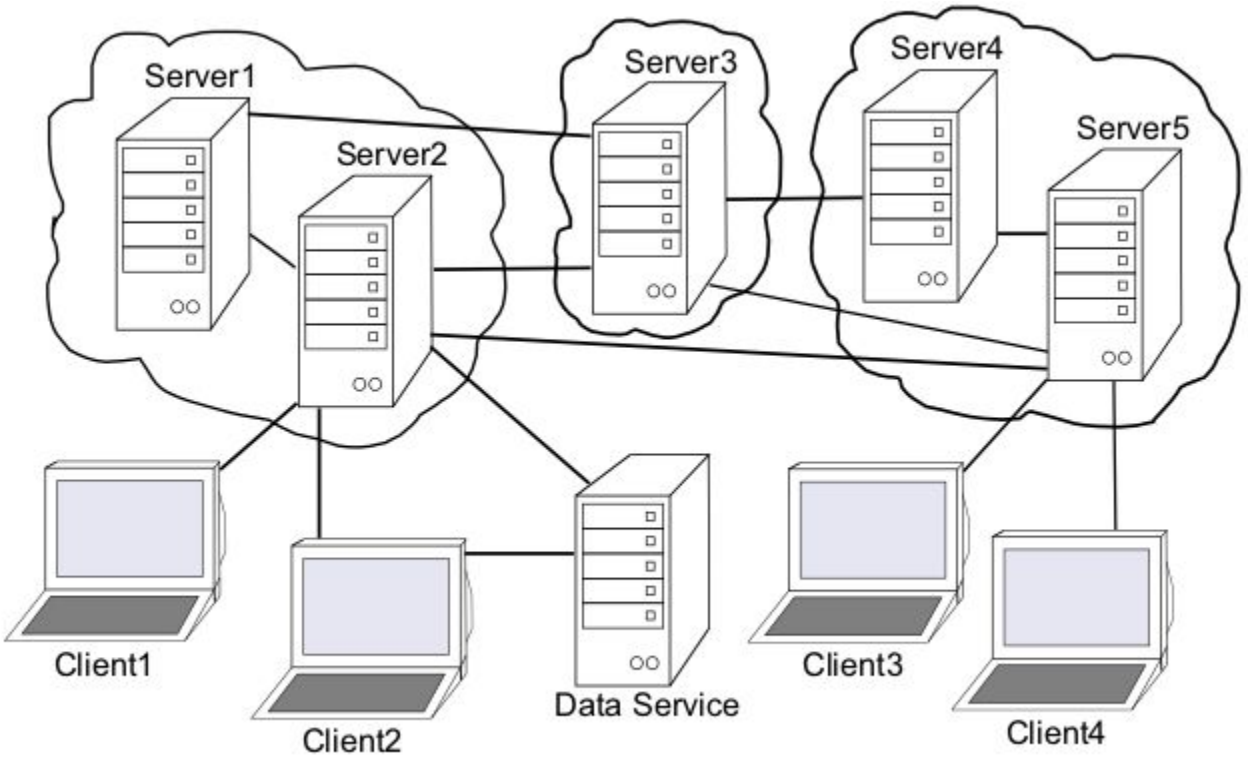
The Core Features

Part 2.

Distributed Architecture

- Servers scattered over network
  - Potentially big network latency
  - Static world partitioning, that would minimalize the network traffic, is not assumed
  - Fully automatic and transparent object distribution
    - Without the need for manual administration
- Three types of „nodes“
  - Simulation servers, clients, data servers

The Massiv Deployment



Security

- Data encryption between all node paris
  - RSA-based authentization
  - Symmetric encryption while transferring data
- Restricted rights of client nodes
  - Clients can affect simulation state indirectly only, via sending requests to special objects

Object Model

- „Managed objects“
  - Objects are automatically managed by the system
  - The classes are written in C++
    - Special coding instructions are defined
    - Special data types
  - Classes described in IDL (Interface Definition Language)
    - Serialization, introspection, RPC
  - Local objects can be accessed directly
- Local garbage collector

Migrations

- Any object is „owned“ exactly by one node
- Object migrations
  - Primary way of object collaboration
    - Migrations are addressed by objects
    - Object = message
  - Primary way to control simulation
    - Simulation is driven by events
    - Migration delivery = event

Replication

- Usage:
  - Data transfer due to simulation presentation to client nodes
  - Reduction of network traffic among servers
- An object can be replicated to an arbitrary node set
  - Read-only copy of an object (part)
  - Automatically kept in a consistent state

Migration And Replication Groups

- Migration and replication groups of objects are always handled as a whole
- Objects belonging to the same migration group are local with respect to each other
  - Primary way to ensure effeciency in a distributed environment with a big network latency
- Group membership is dynamic
  - driven by references to objects:
    - Dynamic data structures (double-linked list)
    - Player character together with its inventory

Remote Procedure Call

- Implemented on top of object migrations
- Asynchronous RPC
  - Delivery can be scheduled to a particular simulation time
  - Ability to retreive call results (polling)
- Synchronous RPC
  - Does not block deliveries of other events

Other Core Features

- Transparent archivation of the simulation state (always consistent)
  - Does not have a negative impact on simulation smoothness
- Data download on the background
  - Data can be updated online
  - Data stored in a tree-like structure
  - Usage: configuration, presentation content (textures, models)
- Server load balancing

The Core Features In The Demo

Part 3.

The Demo

- An exemplary 3D application
- Uses all features provided by the Core
- Examples of basic problems, that have to be dealt with while writing an online distributed game, principles and their solutions in the Massiv environment
- Not a basis for a real online game

3D Map

- Virtual world consists of a map, the players move on
- The map is split to rectangular sectors
  - Each sector can be owned by a different server
- Sectors are not data objects but managed objects
  - Terrain modification in the real time (hills, valleys)
  - The changes are presented to clients by object replication

Sector (1)

- Elevation map (hills, valleys)
  - Divided to 8x8 rectangular tile map
  - Each tile has its own properties (grass, rock, ...)
- Owns entities
  - Moveable: player characters, sheep
  - Decorations: trees, grass, buildings

Sector (2)

- Sector and all owned entities form a single migration group
  - All operations done within the boundaries of a single sector are local and fast
  - Sector can directly manipulate with owned entities
  - Sector migration to a remote server transfers owned entities as well

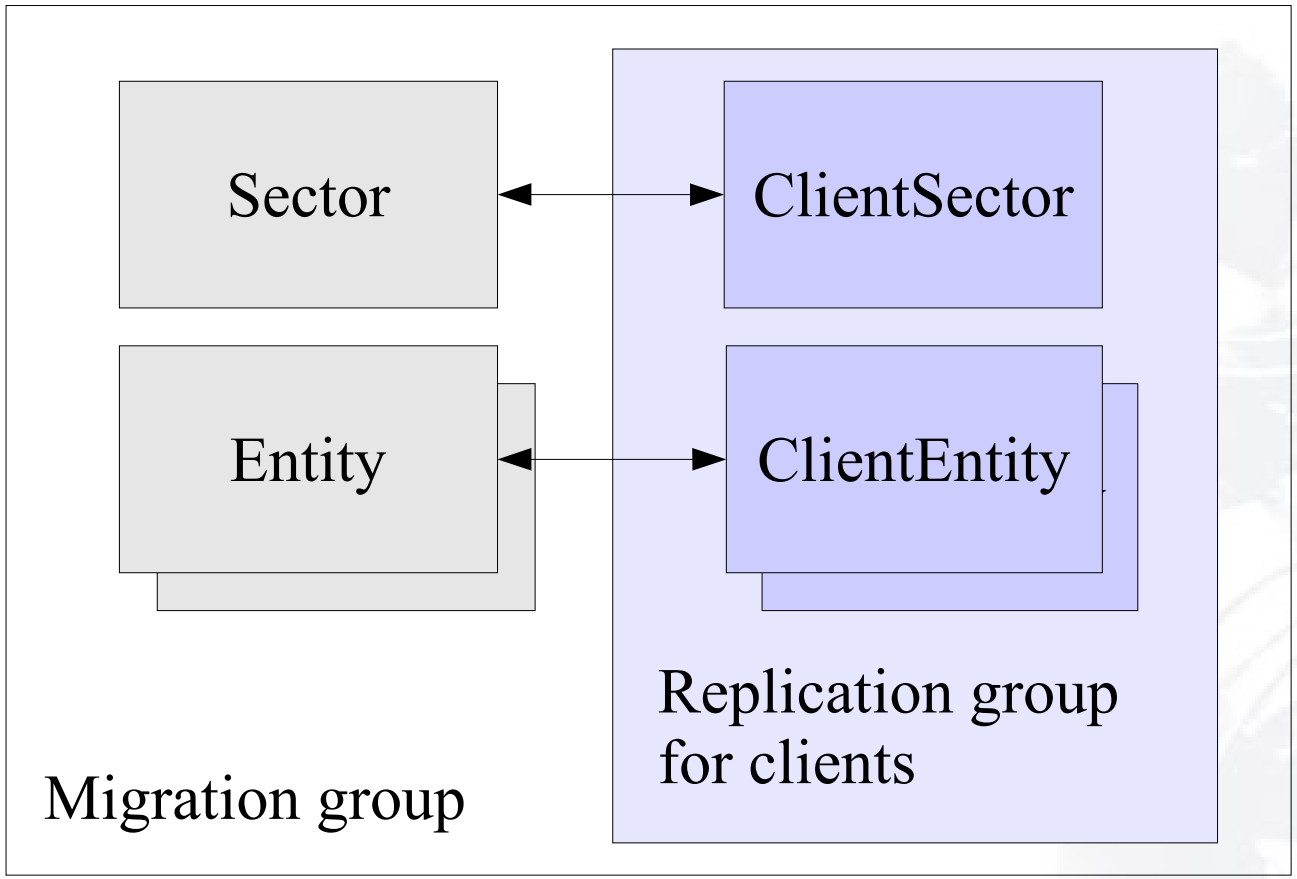
Moving Entities

- Inside the sector like in a non-distributed application
- To a different sector
  - The entity is unlinked from the current sector
    - The migration group of the sector is split
  - The entity migrates to the other sector
  - Once delivered the entity is linked to the new sector
    - The migration group of the entity is merged with the migration group of the new sector

Replication

- Each game object is implemented by two classes
  - Public class – is replicated to clients
    - Holds data that need be interpreted by clients to present the simulation state to users
  - Private class
    - Internal object logic
- This separation allows to
  - Transfer the minimum data to clients
  - Improve security
    - Client applications do not see structure, nor contents, of server-only objects

Sectors And Entities



Project Evaluation

Part 4.

Advantages And Disadvantages Of The Object Model

- Pros
  - Use of C++ – high effeciency
  - Generality of the model
  - Not limited to online games only
- Cons
  - Because of potential high network latency among servers the most recent copies of objects are not available – if a server crashes the application must be restarted from the latest archive

What Went A Treat

- Abstract model of messaging
  - No difference between an object and a message
  - An easy implementation of RPC
- The full use of C++ and STL
- Many additional features that were not originally planned to implement
  - Synchronous RPC and managed exceptions
  - Garbage collector

What Did Not

- Innacurate Draft
  - A lot of changes during the development stage
- Debugging
  - Standard (local) debugging techniques can not be used to debug distributed systems
- Many additional features that were not originally planned

The Demo Presentation

Part 5.

The Demo Presentation

- Data download
- Motion prediction
- Editor
- Console
- Chat

Data download

- All static data (textures, models) implemented as data objects
  - Allows to run clients with no (preinstalled) data
  - Data are downloaded when they are needed (the download speed can be set up)
  - When certain data object is not available (has not been downloaded yet) a proxy object is used.

Editor

- Allows to do online:
  - Edit height map (model terrain)
  - Modify terrain properties (materials)
  - Add, delete and move entities
  - Change object properties (decoration models)
- Editor is accessible to privileged clients only (administrators)

Motion Prediction

- Data from servers transferred to clients infrequently (several times per second only
- Client predicts entity motion in order to improve visual impression (motion smoothness)

Entity Replication

- Public part: ClientEntity
  - Position in the map
  - Type (player character, sheep, decoration, ...)
  - Model tags
    - Determines what model will be drawn to present the entity
- Private part: Entity
  - Linkage to the current sector
  - Data used for entity navigation

Sector Replication

- Public part: ClientSector
  - Elevation map and tile materials
  - Explict list of owned entities is not required
    - Entities are replicated automatically because they belong to ClientSector's replication group
- Private part: Sector
  - Explicit list of owned entities
- Only map portions are replicated to clients
  - Periodic requests to replicate sectors in the player's neighborhood; cancelled when the client disconnects