goals.org

Tomas Herman

2012-04-04 Wed

Contents

1 Goals

In this part of the paper I will talk about goals of the project I will be working on. In the first part, I will reason about why I chose the goals the way I chose them while in the second part I will provide a brief summary of the goals in form of a list.

1.1 Reasoning

I wanted to make this project a learning experience, which affected a lot of the decisions about which technologies to use as well as what subset of functionality described in the part about Minecraft to implemented.

Reader would hopefully agree that while Minecraft is based on quite simple ideas, it is still a complex universe with a lot of details to implement. I wanted to focus mainly on basics, which I thought at the time would be most important for further development in the future.

1.1.1 Server related goals

The main focus of this project is the server infrastructure, which I hoped would be very independent of Minecraft itself. If I would have had done my work correctly, Specus server could be used for any other game or project easily.

- Simplicity
  I wanted the server infrastructure to be very simple to use and simple to reason about, because as I learned in my previous projects, building concurrent systems with networking IO can be quite difficult to get right. In order to achieve that, I used Scala programming language, which is said to have great support for concurrent programming.

  I also decided to use Actor pattern, which seemed very interesting and very natural to use for when dealing with concurrency.

  I also wanted to abstract away all the IO operations and the lower level mechanics of the server. I didn't want to deal with no buffers, sockets or channels when working on business logic.

- Extensibility
  From the description of Minecraft above, I hope it is clear to the reader how important, fun and interesting are the Minecraft extensions. That is why I wanted my server to be built with extensions in mind from the start.

  I wanted it's extension system to be powerful enough to be able to implement entire Minecraft business logic as extension (extensions are called Plugins later in the text and in the code).

  I wanted plugin programmers to be able to express dependencies on other plugin, as it's very common use case that a plugin wants to extend or cooperate with functionality provided by other plugins.

- Distributivity
  I wanted my server to be able to spread the workload into multiple machines, because Minecraft it self has quite big problems with the workload. As mentioned above, map can contain up to $2^{(32+32+8)}$ blocks, so I felt it was important to be able to save all these data into remote database (or cluster of databases).

1.1.2 Minecraft related goals

Because I felt like I chose quite ambitious goals for the server architecture, I decided to keep it simple with the actual logic implementation and treat the Minecraft business logic as a proof of concept. I decided, for now,to only implement just the creative mode described above. That allowed me to skip the implementation of inventory management and monsters, which would take a lot of time.

I also decided to not implement any complicated map generator. I implemented a very simple one for testing purposes which generates simple flat stone world.

I decided not to implement in game maps, signs and items that required any special handling.

I wanted to implement map streaming and on-the-fly map generation, map updates when player makes a change and persistent player position (position of a player is persisted between sessions).

While that is not very impressive set of features, it should provide and test all the important features of the server architecture, as it needs to make a lot of database queries and updates.

1.1.3 Brief summary of goals

Following is the brief summary of the goals mentioned above in form of a list:

- server architecture requirements
  - implemented in Scala
  - extensive usage of Actor mode
  - database support for storing state
  - extensible via plugins
    - must be able to express dependencies on given plugin and it's version
    - must be powerful enough to be able to express entire Minecraft logic
  - IO and socket networking abstracted away
  - state moved from local variables into remote database
- Mincraft functionality requirements

- player position persistence
- on the fly map generation
- map streaming
- map updates by player
- implemented as plugin
- must store all the state in a remote database