Active Object

The Active Object pattern separates the member function execution from the member function call.

- Each object has its thread.
- A member function call is stored in a queue (activation list).
- A scheduler triggers the member function call.

Active Object

Proxy

- Proxy for the member functions on the Active Object
- Triggers the construction of a request object which goes to the Activation List and returns a future
- It runs in the client thread

Member Function Request

- Includes all context information to be executed later

Activation List

- Has the pending requests objects
- Decouples the client from the Active Object thread

Scheduler

- Runs in the thread of the Active Object
- Decides witch request from the Activation List is executed

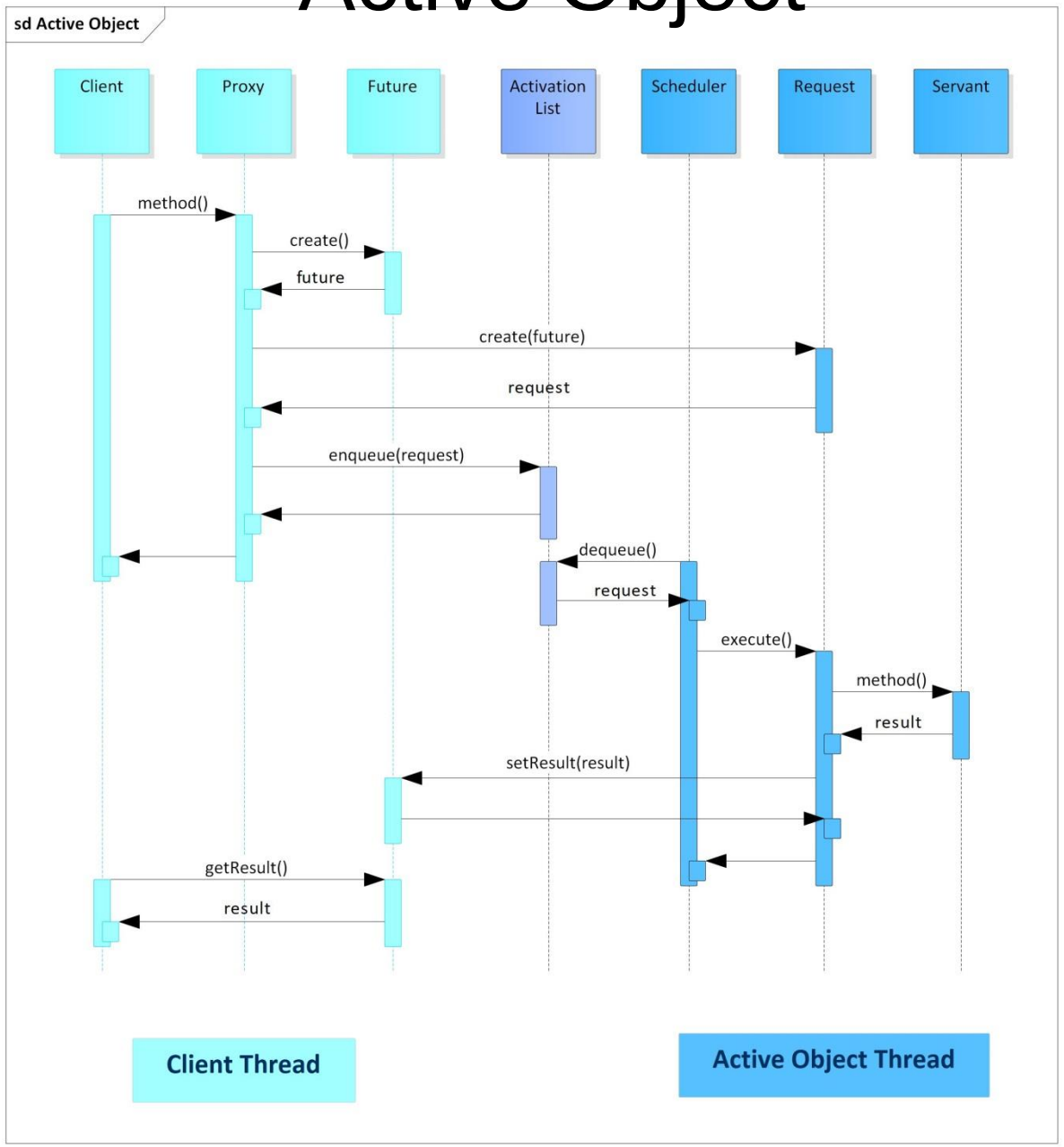
Active Object

- Servant
  - Implements the member function of the Active Object
  - Supports the interface of the Proxy

- Future
  - Is created by the Proxy.
  - Is only necessary if the request object returns a result
  - The client uses the Future to get the result of the request object

activeObject.cpp

Active Object



Active Object

Advantages:

- Only the access to the Active Object must be synchronized
- Clear separation between client and server
- Improved system throughput due to asynchronous execution
- The scheduler can implement different execution strategies for member function processing.

Disadvantages:

- If the member function calls are too fine-granular, relatively high overhead results from the indirection
- Due to the asynchronous member function processing and the different execution strategies, the system can be difficult to debug