Asynchronous Data Provider

Wojciech Koperny, Sławomir Cielepak

December 31, 2020

1 System overview

The Asynchronous Data Provider system is a generic service that acts as a proxy to a simple key-value database. Each entry in the database is indexed with unique Id. The backend that will handle such storage is yet to be decided.

A provider exposes a C++ interface that can be used by external applications directly in the code. It is provided as a shared or static library.

The main goal of the system is to provide concurrent access to the data for multiple threads, thus maintaining the high responsiveness of the system.

Each data request is cached in the system and in case of cache miss, a request is passed on a TaskPool queue. Tasks are consumed in parallel and provide results asynchronously directly to the client.

It is assumed that retrieval of data from the database might be occupied with high resource usage and be time consuming.

1.1 System objects

The figure 1 shows the main system objects.

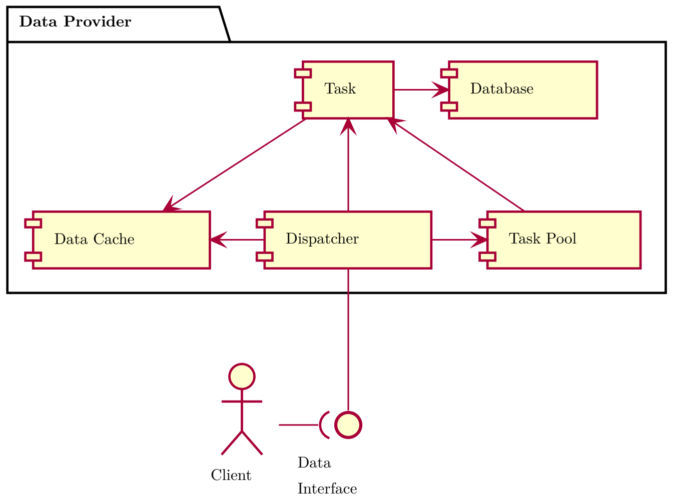

Figure 1: Asynchronous Data Provider system overview

1.1.1 Dispatcher

A Dispatcher is an entity that provides boundary interface for the system receiving the Data Requests form the Client. For every request a `std::future<T>` object is returned.

The Dispatcher first tries to get the requested data from Cache, if it's available, it sets the `std::promise<T>` immediately. In case the data is not available, Dispatcher creates a new Task and puts in on the TaskPool queue.

1.1.2 Data Cache

<description>

1.1.3 Task Pool

<description>

1.1.4 Database

<description>

2 Use cases

2.0.1 Client requests data by ID

<description>

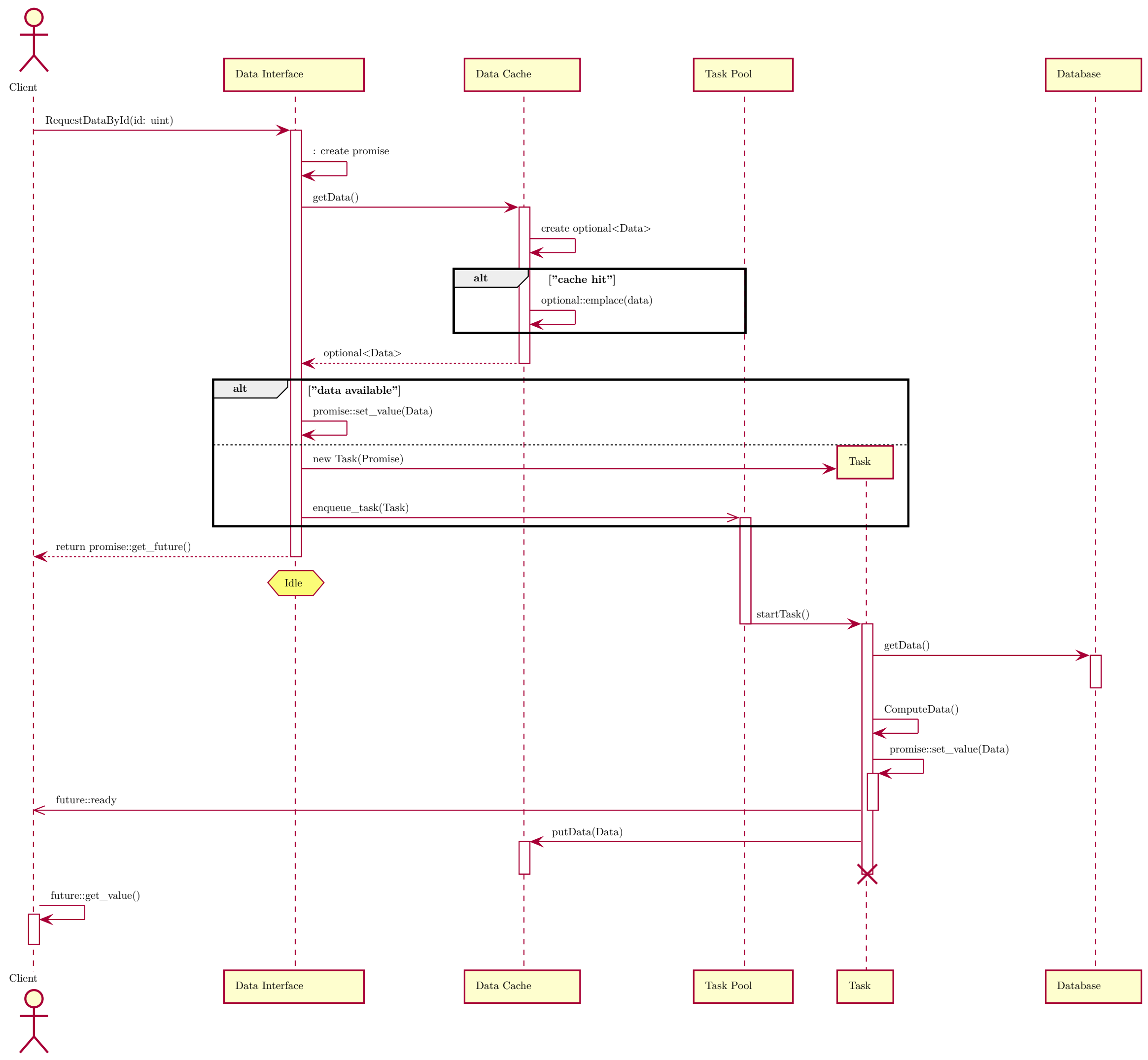

Figure 2: PlantUML diagram from file

3 Interfaces

<description>

```
#include <iostream>

int main()
{
    std::cout << "Hello World\n";
}
```

Listing 1: Example source code from external file