Proposal for a Data Management API within the GridRPC

Y. Caniou, E. Caron, F. Desprez, G. Le Mahec

October 25, 2007

Data Management in the GridRPC

Aims of the Data Management API

- To avoid useless transfers of data
- Generic API unrelated to the data, location of the data, access protocol, etc.
  - $\longrightarrow$ Transparent access to the data from the user point of view
- Homogeneous use of different data transfer protocols
- To improve interoperability between different implementations
- To give an answer to the Saga Working Group

Data Management in the GridRPC

Constraints

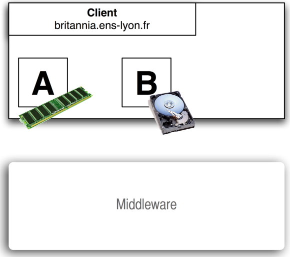
- Must be an optional improvement of GridRPC applications
- Must be in accordance with the GridRPC API
- Should be extensible to existent and future data transfer protocols
- Should unify the access to the data regardless of their sources, types and transfer protocols

Simple RPC call with input and output data

- Data A and B are stored on the client
- One server provides the “*” service
- Result C has to be sent back to the client

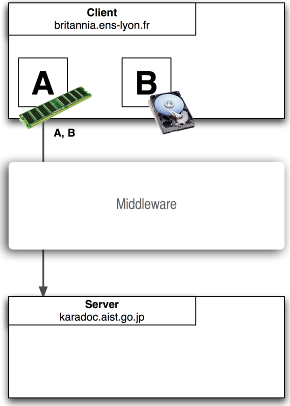
Simple RPC call with input and output data

A in memory / B on disk



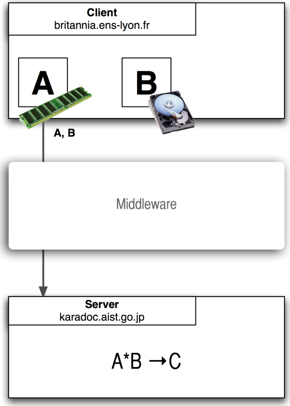
Simple RPC call with input and output data

A and B are transfered to the server



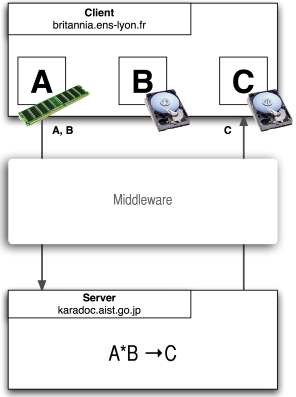
Simple RPC call with input and output data

Computational step



Simple RPC call with input and output data

C is sent back to the client

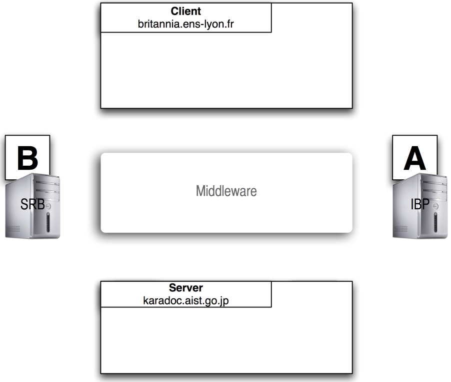


External storage resources

- Data A is stored on a IBP server on the grid
- Data B is stored on a SRB server on the grid
- Data A has to be stored on the client
- Data B has to be stored on the IBP server
- Result C has to be sent back to the client

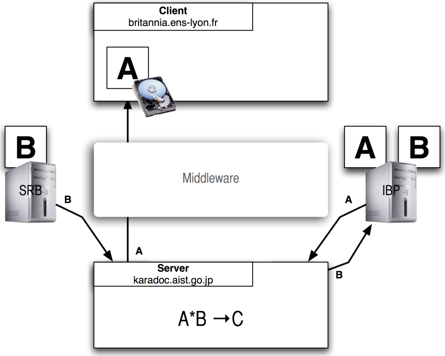
External storage resources

A on IBP server / B on SRB server



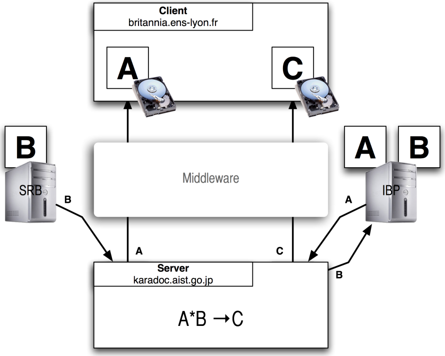
External storage resources

Data are transfered following the input/output rules described in the call + Computational step



External storage resources

C sent back to the client

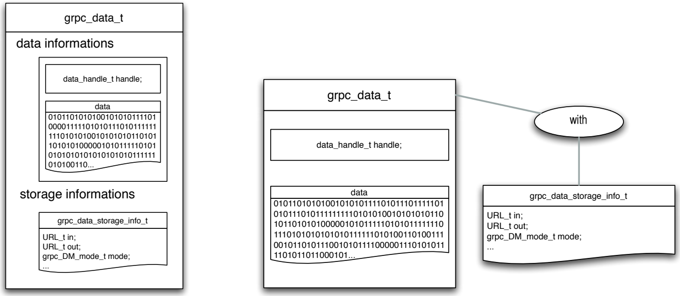


The proposed API defines:

- 2 data structures
- 7 functions

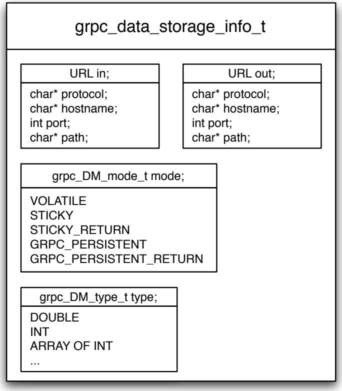
GRPC data type

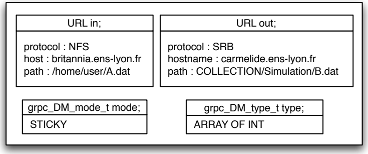
The *grpc_data_t* type contains the data or a handle on it.
The *grpc_data_storage_info_t* of a data can be in the *grpc_data_t* structure or transmitted separately to the middleware.



GRPC data storage information type

The *grpc_data_storage_info_t* type contains the URL where the data can be accessed, the URL where the data will be sent after the call, the management mode and the type of the data.





Data Management Functions

The **grpc_data_init** function

```
grpc_error_t grpc_data_init(grpc_data_t* data,
                            char* URL_input,
                            char* URL_output,
                            grpc_DM_type_t variable_type,
                            grpc_DM_mode_t storage_mode);
```

This function initializes the GridRPC data with a specific data.

Data Management Functions

The **grpc_data_write** function

```
grpc_error_t grpc_data_write(grpc_data_t data,
                             <char* server_name>);
```

This function writes a GridRPC data to the output location set during the init call. A list of additional servers on which the data has to be uploaded can be provided.

Data Management Functions

The grpc_data_read function

```
grpc_error_t grpc_data_read(grpc_data_t* data);
```

After calling the grpc_data_read function, the data will be available in the GridRPC data type data, which will also still contain the Data Handle.

Data Management Functions

The grpc_data_free function

```
grpc_error_t grpc_data_free(grpc_data_t data);
```

After calling the grpc_data_free function, *data* does not reference a GridRPC data. This function may be used to explicitly erase the data on a storage resource.

Data Management Functions

The **grpc_data_getinfo** function

```
grpc_error_t grpc_data_getinfo(grpc_data_t data,
                               grpc_data_info_type info
                               char* info);
```

This function let the user access information about the grpc_data_t. It returns information on data characteristics, status, and location.

Data Management Functions

The **grpc_data_load** and **grpc_data_save** functions

```
grpc_error_t grpc_data_load(grpc_data_t data,
                            char* buffer);
grpc_error_t grpc_data_save(grpc_data_t data,
                            char* buffer);
```

These functions are used to save/load the necessary informations to use the data stored on the grid.

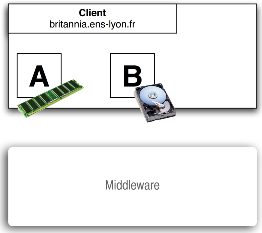
Data Management Functions

The **grpc_error_t** type possible values

Error code identifier	Meaning
GRPC_NO_ERROR	Success
GRPC_INVALID_TYPE	Specified type is not valid
GRPC_INVALID_MODE	Specified location is not valid
GRPC_OTHER_ERROR_CODE	Internal error detected

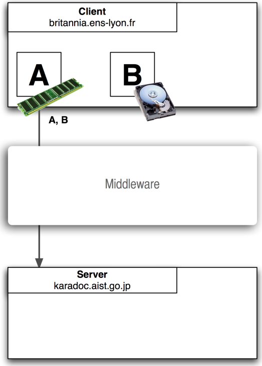
Simple RPC call with input and output data

```
grpc_data_init(&dhA, "LOCAL_MEMORY://britannia.ens-lyon.fr/&A", NULL, DOUBLE, NULL);
grpc_data_init(&dhB, "NFS://britannia.ens-lyon.fr/home/user/B.dat", NULL, DOUBLE, NULL);
grpc_data_init(&dhC, NULL, "NFS://britannia.ens-lyon.fr/home/user/C.out", DOUBLE, NULL);
```



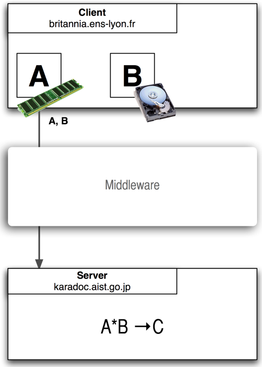
Simple RPC call with input and output data

grpc_function_handle_init(handle1, "karadoc.aist.go.jp", "*");



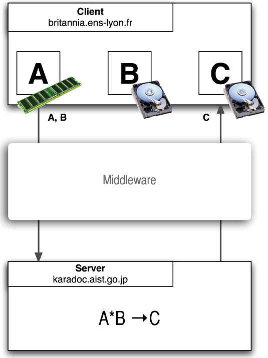
Simple RPC call with input and output data

grpc_call(handle1, dhA, dhB, &dhC);



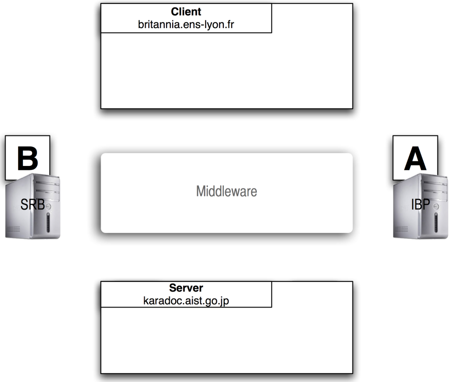
Simple RPC call with input and output data

Output data C is sent back to the client.



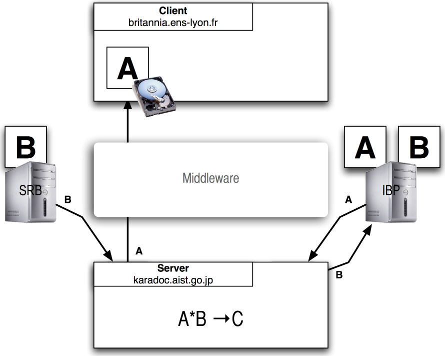
Storage with external storage resources

```
grpc_data_init(&dhA, "IBP://kaamelott.cs.utk.edu/1212#A.dat/ReadKey/READ",
               "NFS://britannia.ens-lyon.fr/home/user/A.dat", DOUBLE, NULL);
grpc_data_init(&dhB, "SRB://carmelide.ens-lyon.fr/COLLECTION/Simulations/B.dat",
               "IBP://kaamelott.cs.utk.edu/1213#B.dat/WriteKey/WRITE", DOUBLE, NULL);
grpc_data_init(&dhC, NULL, "NFS://britannia.ens-lyon.fr/home/user/C.out", DOUBLE, NULL);
```



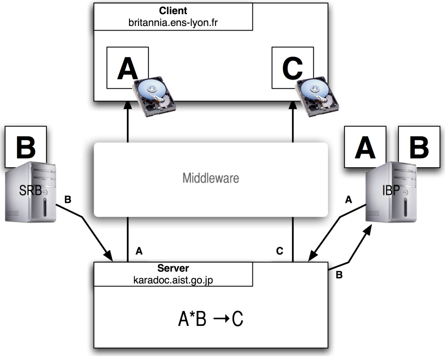
Storage with external storage resources

grpc_call(handle1, dhA, dhB, &dhC);



Storage with external storage resources

Output data C is sent back to the client.



Conclusion & future works

- Simple API for data management with only 7 functions
- Allowing a simple and powerful data management from the API
- Taking into account many use cases (all?)

- How to manage multiple data repositories?
- Implementation
- GridRPC data management interoperability
    - New document
    - Interoperability testing for the GridRPC data API specification
    - Error codes to be defined