A Remote Procedure Call Approach for Extreme-scale Services

Jerome Soumagne[1], Philip Carns[2], Dries Kimpe[3], Quincey Koziol[1], and Robert Ross[2]

[1]The HDF Group
[2]Argonne National Laboratory
[3]KCG

Introduction

When working at exascale, the various constraints imposed by the extreme scale of the system bring new challenges for application users and software/middleware developers. In that context, and to provide best performance, resiliency and energy efficiency, software may be provided as a service oriented approach, adjusting resource utilization to best meet facility and user requirements. These services, which can offer various capabilities, may be used on demand by a broad range of applications.

Remote procedure call (RPC) (1) is a technique that originally followed a client/server model and allowed local calls to be transparently executed on remote resources. RPC consists of sending local function parameters to a remote target that executes the corresponding function call, returning the result back to the caller. Reusable services require a communication method in order to be remotely accessed and for this purpose, RPC can serve as a foundation. We introduce a building block that enables this ecosystem for software and middleware developers with an RPC framework called **Mercury** (2).

RPC for High-Performance Computing

Mercury is designed for high-performance computing systems. It takes advantage of **native high-speed interconnects** and exposes the semantics required for making **nonblocking RPC** as well as for supporting **large data arguments**.

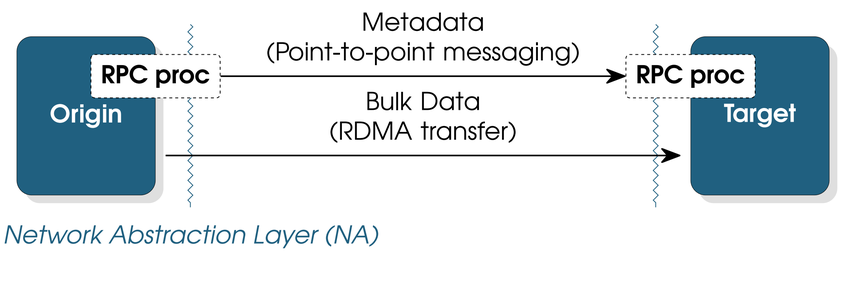


Fig. 1: Mercury defines two separate layers for metadata and bulk data transfers.

Basis for Reusable Services

To serve as a basis for accessing and enabling reusable services in a high-performance computing environment, Mercury is designed to be both easily integrated and extended by having client and server concepts abstracted by the notions of **origin** and **target**, since a server may also act as a client, and vice versa.

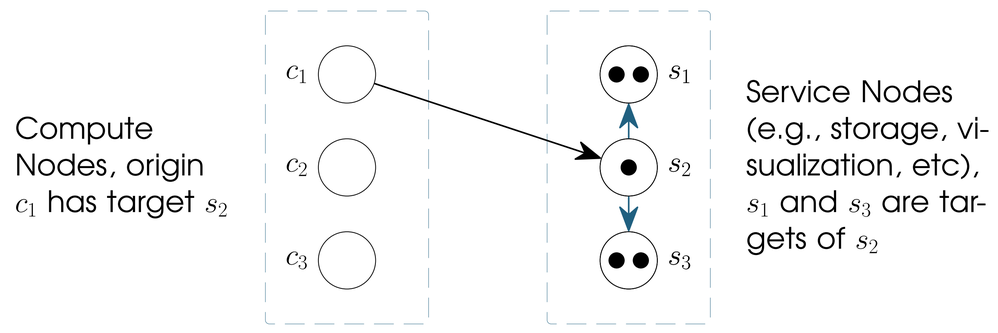
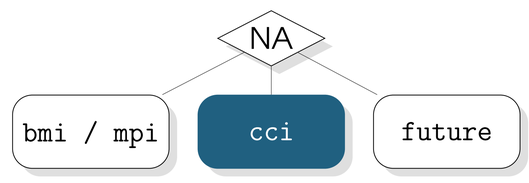


Fig. 2: Mercury uses *origin* and *target* concepts for easy integration within distributed services.

Support for HPC Transports

Mercury provides a **network plugin mechanism** that can support existing as well as future network fabrics. The network abstraction layer requires only a minimal necessary set of functionality and therefore makes it easy for developers to create a new plugin.



`bmi / mpi`: TCP/IP, IB
`cci`: TCP/IP, IB, Cray GNI, shared memory

Fig. 3: Mercury can support existing as well as future network fabrics through NA plugins.

Enable High Concurrency

To enable high concurrency, the Mercury progress and execution model is based on a **callback model**, as opposed to a traditional request-based model.

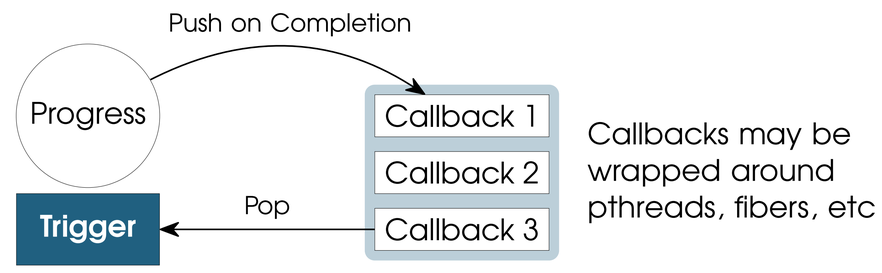


Fig. 4: Mercury's callback model allows flexible and non-blocking execution flows.

This has two advantages: first it allows upper layer services built on top of Mercury to easily schedule operations by using for instance, a **multi-threaded execution** model; second, it still allows definition, when necessary and more convenient, of **shim layers** that simplify common cases, based for instance on a request model to provide post/test operations.

Upcoming Challenges

Defining reusable software services at exascale is a challenge. For such, Mercury will be a valuable asset and serve as a basis by providing a lightweight and modular RPC infrastructure for high-performance computing middleware, enabling both high-speed transfers and high concurrency.

Higher-level features such as multithreaded execution, pipelined operations, or other auxiliary features such as group membership, authorization, etc, are not provided by Mercury directly, although Mercury is designed to provide the ecosystem so that these features can easily be built on top of it.

References

(1) A. D. Birrell and B. J. Nelson. Implementing Remote Procedure Calls. *ACM Trans. Comput. Syst.*, 2(1):39–59, 1984.

(2) J. Soumagne, D. Kimpe, J. Zounmevo, M. Chaarawi, Q. Koziol, A. Afsahi, and R. Ross. Mercury: Enabling Remote Procedure Call for High-Performance Computing. In *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–8, Sept 2013.

See Also

For more details about the Mercury project, please visit the project page:
`http://www.mcs.anl.gov/projects/mercury`
Or the GitHub page:
`http://github.com/mercury-hpc/mercury`