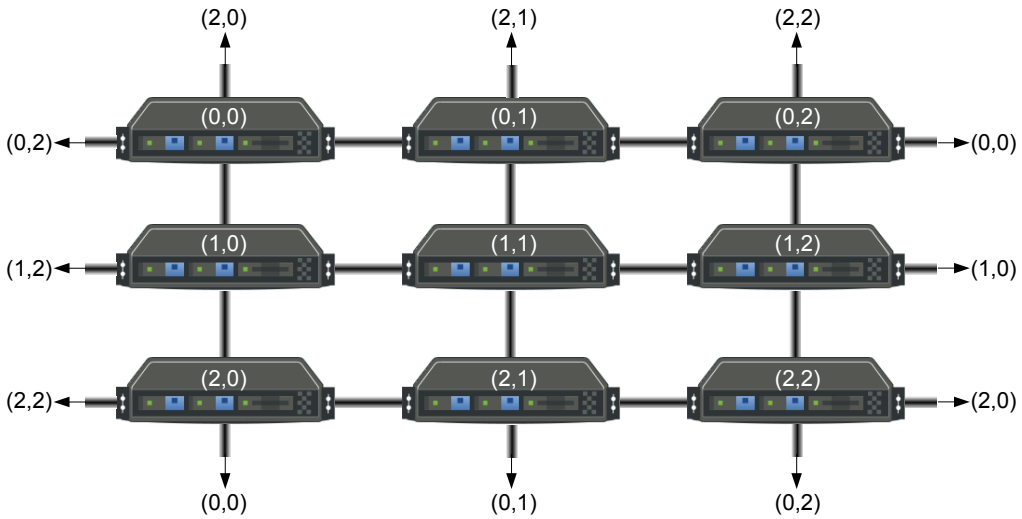

Figure 1.2.: Network topology of a 2D-Torus. Leftmost nodes (X,0) are interconnected with the rightmost nodes (X,2). Top nodes (0,X) are connected with the bottom nodes (2,X).

send a message to the center node (1,1). The routing algorithm might route both messages via node (0,1); however, the node has only one link to node (1,1).

Another frequently deployed class of network topologies are *hierarchical networks* which add intermediate layers of hardware that relay the data; a switch connects many links and forwards incoming data into the direction of the intended receiver. This approach provides a higher aggregated bandwidth between the nodes, but increases the latency of the communication. One hierarchical network topology is a *Clos* network, where a high number of links between switches offers the maximum available network bandwidth between all connected hosts. This network topology is often deployed with Infiniband technology. Refer to [AK11] for more details on network technologies and topologies.

Storage Data management in bigger clusters is performed by distributed file systems, which scale with the demands of the users. Usually, multiple file systems are deployed to deal with disjoint requirements.

In a typical setup two file systems are provided: a fast and large scratch space to store temporary results, and a slower, but highly available volume to store user data (e.g., for home directories). Looking at a particular distributed file system, a set of storage servers provides a high-level interface to manipulate objects of the namespace. The *namespace* is the logical folder and file structure the user can interact with; typically it is structured in a hierarchy. Files are split into parts which are distributed on multiple resources and which can be accessed concurrently to circumvent the bottleneck of a single resource. Replication of a part on multiple servers increases availability in case of server failure. Truly *parallel file systems* support concurrent access to disjoint parts of a file; in contrast, conventional file systems serialize I/O to some extent[10].

An example of the *hierarchical namespace* and its mapping to servers is given in Figure 1.3. The directory "home" links to two subfolders which contain some files. In this case, directories are mapped to exactly one server each, while the data of logical files is maintained on all three servers. Data of "*myFile.xyz*" is split into ranges of equal size and these blocks are distributed round-robin among the data servers. Each server holds three ranges of the file.

Storage devices are required to maintain the state of the file system in a persistent and consistent way. A storage device can be either directly attached to a server (e.g., a built in hard disk), or the server controls devices attached to the network. In contrast to the interface provided by file systems, the interface to

[10]More details on file systems in HPC systems are provided in Section 2.1.

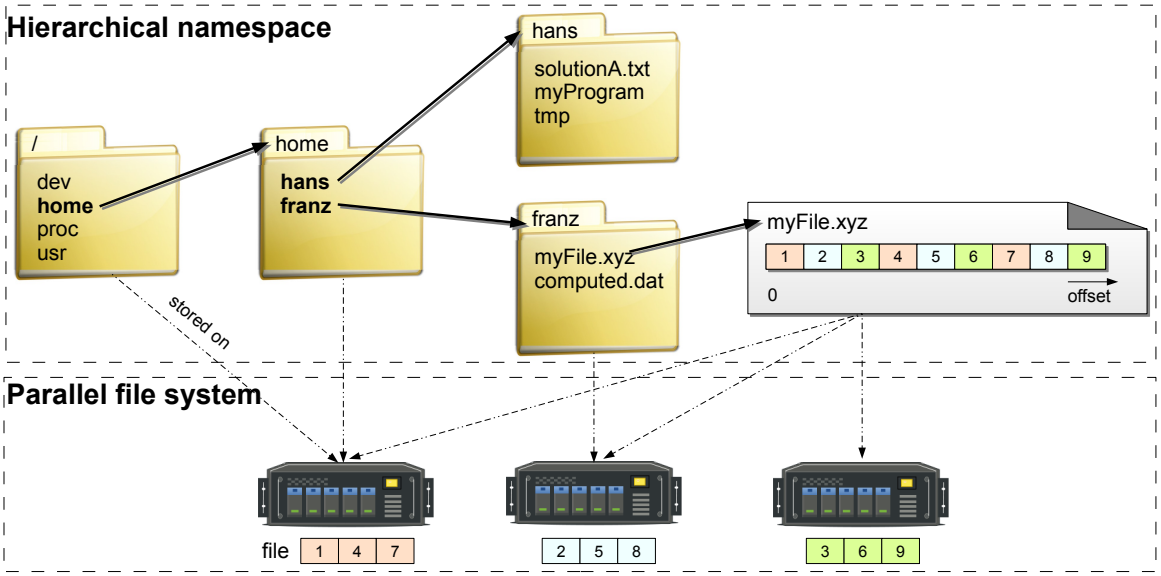

Figure 1.3.: Example hierarchical namespace and mapping of the objects to servers of a parallel file system. Here, metadata of a single logical object belongs to exactly one server, while file data is distributed across all servers.

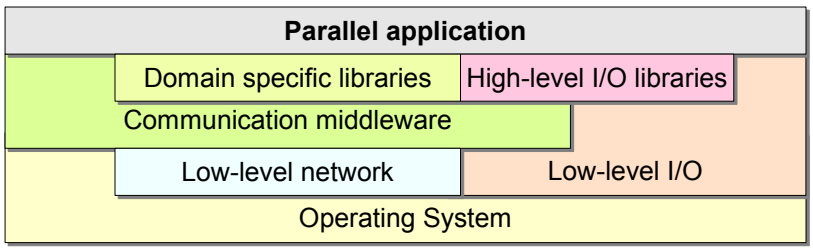

Figure 1.4.: Representative software stack for parallel applications.

storage devices is low-level. At the block-level, data can be accessed with a granularity of full blocks by specifying the block number and *access type* (read or write).

In a *Storage Area Network* (*SAN*) the block device seems to be connected directly to the server. To communicate with the remote block storage device, servers use the *Small Computer System Interface* (SCSI) command protocol. A SAN could share the communication infrastructure, or another network just for I/O can be deployed. Fibre Channel and Ethernet are common network technologies with which to build a SAN. In the latter case, SCSI commands are encapsulated into the IP protocol, that is, the so-called *Internet SCSI* (iSCSI). Therefore, the existing communication infrastructure can be used for I/O, too.

1.1.2. Software Layers

Several software layers are involved in running applications on supercomputers. A representative software stack is shown in Figure 1.4.

A *parallel application* uses a domain-specific framework and libraries to perform tasks common to most applications in that field. For example, the numerical library Atlas[11] is widely adopted by scientists.

Since collaboration between remote processes of an application requires communicating data, a conveniently programmable interface and an efficient implementation is important. The service to exchanged data is offered by the *communication middleware*. Several programming paradigms and models exist for

[11] Automatically Tuned Linear Algebra Software