On the Performance and Scalability of Client-Server Based Parallel Disk I/O*

Erich Schikuta, Thomas Fuerle, Kurt Stockinger, and Helmut Wanek
Institute for Applied Computer Science and Information Systems
University of Vienna, Rathausstr. 19/4, A-1010 Vienna, Austria, schiki@ifs.univie.ac.at

In the last few years the applications in high performance computing (Grand Challenges) shifted from being CPU-bound to be I/O-bound. Performance can not be scaled up by increasing the number of CPUs any more, but by increasing the bandwidth of the I/O subsystem. This situation is commonly known as the I/O bottleneck in high performance computing We see a solution to this problem in a dedicated, smart, concurrent executing runtime system, gathering all available information of the application process both during the compilation process and the runtime execution. This approach led to the design and development of the Vienna Input Output System, ViPIOS.

The ViPIOS System

The Vienna Parallel Input Output System (ViPIOS) is an I/O runtime system, which provides efficient access to persistent files, by optimizing the data layout on the disks and allowing parallel read/write operations. ViPIOS is targeted as a supporting I/O module for high performance languages (e.g. HPF). ViPIOS was designed with data engineering know-how in mind. The driving design principles were performance and scalability. Performance demands coordinated parallel data accesses of processes to multiple disks. Scalability guarantees that the size of the used I/O system is defined by or correlated with the problem size only.

The ViPIOS system architecture is based on a client-server approach, consisting of a set of cooperating server processes, which accomplish the requests of the application client processes by data accesses to the respective disk sub-system (see figure 1).

A ViPIOS server process consists of several functional units organized in 3 layers: The Interface layer provides the connection to the "outside world"(i.e. applications, programmers, compilers, etc.). Different interfaces are supported by *interface modules* to allow flexibility and extendability. Until now we implemented an HPF interface module (aiming for the VFC, the HPF derivative of Vienna FORTRAN) a (basic) MPI-IO interface module, and the specific ViPIOS interface which is also the interface for the specialized modules. The Kernel layer is responsible for all server specific tasks. The Disk Manager layer provides the access to the available and supported disk sub-systems. Also this layer is modularized to allow extensibility and to simplify the porting of the system. Available are modules for ADIO, MPI-IO, and Unix style file systems.

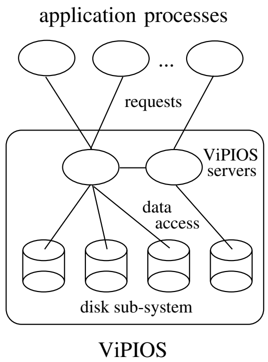

Figure 1: ViPIOS system architecture

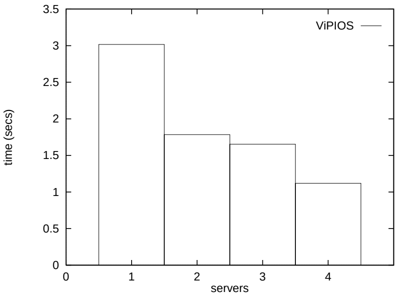

Figure 2: Scalability results

The Performance of ViPIOS

We compared the available ViPIOS implementation to standard UNIX I/O and to MPI-I/O on a workstation cluster environment. Our tests were set up in a way to inhibit any file access optimizations (like raw file access, caching or data redistribution). This way we could prove that the overhead posed by ViPIOS is reasonably small. The scalability behavior is depicted by figure 2. Regarding the promising results of our first performance tests presented in this paper we expect ViPIOS to outperform other existing systems.

*This work was carried out as part of the research project "Language, Compiler, and Advanced Data Structure Support for Parallel I/O Operations" supported by the Austrian Science Foundation (FWF Grant P11006-MAT)