Fast Serial Append File I/O Mode For Cilk

Alexandru Caracaş
CSAIL MIT

***Abstract*—We introduce Serial Append, a new parallel file I/O mode well suited for parallel applications. Serial Append allows parallel applications to efficiently perform file I/O in parallel while maintaining a structure that makes the sequential file I/O operations easily available. This property of Serial Append lets serial applications, that are strictly dependent on sequential file I/O operation, to be easily parallelized. Serial Append can be used for example to parallelize compression utilities such as bz2, or to add logging to existent parallel applications, or as additional support to database applications. We present a cost efficient algorithm for performing Serial Append that uses concurrent Skip-Lists to guarantee write as well as read and seek file operations. We discuss the implementation of the algorithm for the Cilk multithreaded language. We also show the performance of several serial applications that were ported to use Serial Append.**

As an example of Serial Append consider a parallel execution (on several processors) of a multithreaded computation that performs output to a file. The output of the parallel execution of the program is scrambled, namely the output of one processor is interleaved with the output of the others. This does not correspond to the sequential (single processor) execution of the multithreaded computation. With negligible slow down in the parallel execution, Serial Append allows for the serial output to be easily available and reconstructed by saving the meta-data of the parallel execution of the computation in an additional structure. This permits efficient, parallel access, to the parallel file I/O operations as if they were performed sequentially.

Cilk is a multithreaded language for writing parallel applications, and it has a provably good work-stealing scheduler. To make Serial Append work for Cilk it is important to keep track of the steal operations that occur in the parallel execution of a Cilk program. We define an I/O node to contain all the write operations that a processor performs between two steals. The implementation of Serial Append for Cilk is closely coupled with the Cilk runtime and scheduler.

Meta-data about the parallel computation, namely the I/O nodes that are created when steal operations occur, is saved in a concurrent Skip-List. This is the only synchronization required during the parallel execution. By using a Skip-List, we improve the performance of the read and seek operations. On each steal operation, a new I/O node is created and inserted in the right order into the Skip-List, namely after the I/O node of the processor from which the current processor stole work. Each processor has its own file I/O buffer and all its write operations are performed into this buffer. In practice, an I/O node only contains the relevant meta-data that allows to access and retrieve the data written to the file I/O buffers. This scheme makes possible for all processors to perform parallel file I/O operations in their own buffers and maximizes the performance of write operations.

[Full Text Not Available]