Barrier (computer science)

Threads synchronization primitive

In parallel computing, a **barrier** is a type of synchronization method. A barrier for a group of threads or processes in the source code means any thread/process must stop at this point and cannot proceed until all other threads/processes reach this barrier.

Many collective routines and directive-based parallel languages impose implicit barriers. For example, a parallel *do* loop in Fortran with OpenMP will not be allowed to continue on any thread until the last iteration is completed. This is in case the program relies on the result of the loop immediately after its completion. In message passing, any global communication (such as reduction or scatter) may imply a barrier.

See also

Rendezvous (Plan 9).

Dynamic Barriers

Classic barrier constructs define the set of participating processes/threads statically. This is usually done either at program startup or when a barrier like the Pthreads barrier is instantiated. This restricts the possible applications for which barriers can be used.

To support more dynamic programming paradigms like fork/join parallelism, the sets of participants have to be dynamic. Thus, the set of processes/threads participating in a barrier operation needs to be able to change over time. X10 introduced the concept of *clocks* for that purpose, which provide a dynamic barrier semantic. Building on clocks, *phasers*[1] have been proposed to add even more flexibility to barrier synchronization. With phasers it is possible to express data dependencies between the participating processes explicit to avoid unnecessary over-synchronization.

Processor and compiler barriers

Memory barrier is a class of instructions which cause a processor to enforce an ordering constraint on memory operations issued before and after the barrier instruction.

A **barrier** can also be a high level programming language statement which prevent the compiler from reordering other operations over the barrier statement during optimization passes. Such statements can potentially generate processor barrier instructions. Different classes of barrier exist and may apply to a specific set of operations only.

References

[1] Shirako, J.; Peixotto, D. M.; Sarkar, V.; Scherer, W. N. (2008). "Phasers". *Proceedings of the 22nd annual international conference on Supercomputing - ICS '08*. pp. 277. doi:10.1145/1375527.1375568. ISBN 9781605581583.

Article Sources and Contributors

Barrier (computer science) *Source*: http://en.wikipedia.org/w/index.php?oldid=456343309 *Contributors*: Adrianwn, Alexbbrown, Asparagus, Avicennasis, BWDuncan, Deineka, Headbomb, Imz, Jugones55, Kuszi, Liao, Paul Foxworthy, Wrp103, 9 anonymous edits

License