Ballot Counting for Optimal Binary Prefix Sum

Anthony Skjellum  David Whitaker (Contact: dwhitt@uab.edu)  Purushotham Bangalore
Department of Computer and Information Sciences, University of Alabama at Birmingham

__ballot()__ evaluates predicate for all threads of the warp and returns an integer whose Nth bit is set if and only if predicate evaluates to non-zero for the Nth thread of the warp. This function is only supported by devices of compute capability 2.0.1

- Compiles to a single instruction
- Functions like MPI_ALLTOALL, for a single bit
- Intra-warp communication without shared mem.

__popc(x)___

returns the number of bits that are set to 1 in the binary representation of 32-bit integer parameter x.1

- Compiles to a single instruction on Fermi
- Gives the sum of previous threads in current warp

---

### Application: Radix Sort

The radix sort algorithm1 is among the fastest GPU sorting algorithms currently published. It sorts keys by proceeding from the Least Significant Bit to the Most Significant Bit. At each step, it performs a stable sort of the keys on the current bit only. It accomplishes this by performing a binary prefix sum to determine the number of set bits appearing before each element. Subtracting from the thread index also gives the number of unset bits. Finally, the last thread broadcasts the total number of unset bits to all threads. With this information, each thread can calculate its unique index in the stable order for this bit.

This requires as many Binary Prefix Sums as there are bits in the key value. The Binary Prefix Sum represents the majority of the workload. So, having a faster Binary Prefix Sum yields a significant improvement over current algorithms.

We tested against a simplified version of the Radix Sort implemented in the CUDPP library. Our version removed the multiple element per thread optimizations that would speed up both algorithms equally for simplicity, but otherwise used the same approach.

Our testing reveals an 80% speed improvement over current techniques.

---

### Future Applications: Suffix Array Construction, BWT, BZIP Compression

A suffix array is an array of integers giving the starting positions of suffixes of a string in lexicographical order. Lexicographical sorting must use a slight variation on the theme that Radix Sort employs: it starts from the Most Significant Bit and works downward, but only sorts within the range where the previously sorted bit matches its neighbors. This can be handled by having each thread keep track of the first element that currently matches its own data, as well as the number of such elements. When this number of elements is 1, the element is in the correct position. However, the thread will still have to wait until all the threads in the block have sorted correctly. Some instances can be conceived for which this worst case is easily possible, for instance, a book which has a repeated sentence or paragraph. A data structure should be utilized which keeps active elements separate from sorted elements and uses fewer and fewer blocks as the sorting grows more accurate. In any case, this should prove faster than comparison-based sorting, which would have the same issue with every comparison.

The Burrows Wheeler Transform, which comprises a major step in BZIP compression, is a variation on the suffix array. In fact, if the string is seen as having a null character appended that is ignored when determining suffixes of the string, the Burrows Wheeler Transform is applicable.

This work supported by the National Science Foundation Grant CNS-0821497