

The FG Programming Environment: Good and Good for You

Elena Riccio Davidson *

Dartmouth College Department of Computer Science
laney@cs.dartmouth.edu

Categories and Subject Descriptors

D.1 [Software]: Programming techniques

General Terms

Performance, Design, Human Factors

Keywords

Parallel programming, human factors, usability

FG, a programming environment for parallel programs running on distributed-memory clusters, improves programmer productivity for an asynchronous, pipelined program in two metrics: code-development time and lines of source code. For our test case, FG programmers produced a solution in 29% less time and with 34% fewer lines of source code than non-FG programmers.

Typically, a parallel programmer employs two approaches for mitigating latency in distributed-memory applications. First, a programmer hides low-latency operations, such as local-memory computation, behind high-latency operations, such as disk I/O. Second, a programmer accesses data in blocks in order to amortize the cost of transferring data from disk to memory or among the nodes of a distributed-memory cluster. The code required for both techniques, though usually unrelated to the underlying algorithm the programmer is implementing, often accounts for a great deal of source code and can also be difficult to write and debug. FG supplies all such code, leaving the programmer to write only the code required for the algorithm itself.

We quantify improved productivity with two metrics. The first is *code-development time*: the amount of time required to design, write, and debug a particular program is far less with FG than without. The second metric is lines of source code: FG programs are significantly smaller than equivalent non-FG programs. We conducted three productivity studies, which show that FG reduces code-development time by an average of approximately 29% and source code size by an average of approximately 34% for an asynchronous, pipelined program.

We administered identical productivity studies at three different universities and recruited 23 subjects. We present results from the 17 subjects who completed the study. We split the subjects into two

*Supported in part by DARPA Award W0133940 in collaboration with IBM and in part by National Science Foundation Grant IIS-0326155 in collaboration with the University of Connecticut. Also supported by National Science Foundation Grant EIA-98-02068.

Copyright is held by the author/owner(s).
SPAA'06, July 30–August 2, 2006, Cambridge, Massachusetts, USA.
ACM 1-59593-262-3/06/0007.

groups: FG programmers and threads programmers. We provided each programmer with working code that performs a completely synchronous version of columnsort [2]. The synchronous program runs on a single processor and uses input matrices small enough to fit into main memory. The program repeatedly reads a matrix into a buffer, sorts the data, and then writes the sorted matrix out to disk. The code we provided contained functions for each of the steps of the columnsort algorithm as well as functions to read from and write to the disk. The provided program arranges the functions in a synchronous pipeline and uses only one buffer.

The nine FG programmers displayed a much shorter code-development time, on average, than the eight threads programmers. The threads programmers worked for an average of 256 minutes, and the FG programmers worked for an average of only 182 minutes. The FG programmers, in other words, required 29% less time to produce complete, working code. We see these results even though the threads programmers had an average of 9.13 years of experience coding, compared to the 8-year average of the FG programmers.

We found that experience in coding does not correlate directly with code-development time. Several FG and threads programmers have identical years of experience, and so we can compare them directly. For example, three FG programmers and three threads programmers each have 10 years of experience coding. All three of these FG programmers exhibited shorter code-development time than any of the three equivalent threads programmers. We see similar trends for the FG and threads programmers with seven, eight, and nine years of experience.

In addition to reducing code-development time, our experimental results show that FG also reduces lines of source code required. In addition to our earlier work [1], our productivity studies show that programmers unfamiliar with FG produced less source code than the threads programmers, all of whom had some experience in coding with threads. Indeed, the columnsort programs that use FG are, on average, 34.3% smaller than the programs that use threads directly. The threads programmers produced an average of 586 lines of source code, whereas the FG programmers produced an average of only 385 lines.

1. REFERENCES

- [1] Elena Riccio Davidson and Thomas H. Cormen. The FG programming environment: Reducing source code size for parallel programs running on clusters. In *Second Workshop on Productivity and Performance in High-End Computing (P-PHEC)*, pages 27–34, February 2005.
- [2] Tom Leighton. Tight bounds on the complexity of parallel sorting. *IEEE Transactions on Computers*, C-34(4):344–354, April 1985.