Implementing a Parallel Sparse Matrix-Vector Multiplication Using Dataflow

Brunno F. Goldstein  
PESC – COPPE – UFRJ  
bfgoldstein@cos.ufrj.br

Tiago A. O. Alves  
PESC – COPPE – UFRJ  
tiagoaoa@cos.ufrj.br

Michael Souza  
DEMA – UFC  
michael@ufc.br

Luiz Mariano Carvalho  
DMA – IME – UERJ  
luizmc@ime.uerj.br

Leandro A. J. Marzulo  
DICComp – IME – UERJ  
leandro@ime.uerj.br

Felipe M. G. França  
PESC – COPPE – UFRJ  
felipe@cos.ufrj.br

ABSTRACT

Sparse matrix-vector multiplication (SpMV) is an important operation in sparse linear algebra problems. According to Bell [1], “they represent the dominant cost in many iterative methods for solving large-scale linear systems”. In this paper we propose a dataflow parallel version of the SpMV kernel using Compressed Sparse Row (CSR) storage format. This implementation is executed in a Dataflow Runtime Environment called Trebuchet [3][4][5].

The dataflow model[2] exposes parallelism by essence. Task execution is triggered as soon as the input data is available. A dataflow program can be described as a dataflow graph where nodes represent instructions (or tasks) and edges represent their input/output dependencies. Two instructions can run concurrently if there is no directed path between them and their pieces of data are available. This differs from Von Neumann model where instructions are guided by control flow and their execution is inherently sequential.

Our experimental environment consists of a machine with 4 Quad-Core Intel Core i7-3820 processors running at 3.6GHz and 64 GB RAM. We performed the SpMV kernel in three matrices from University of Florida Sparse Matrix Collection1, listed in Table 1. Figure 1 presents the results for Dataflow (running on Trebuchet), Intel Math Kernel Library (MKL)2 (which uses OpenMP3 in parallel version) implementations.

Results show that with a naive dataflow implementation it is already possible to achieve higher speedups than the ones obtained with Intel MKL. For small matrices the dataflow runtime environment overheads become more evident, but for larger ones, Trebuchet provides better results. In order to improve performance, we intend to implement other SpMV kernels using different storage formats and other approaches to get better use of cache locality[6]. Moreover, since dataflow allows more flexibility in describing application dependencies, we may find more interesting parallelization patterns that are easier to implement and provide smaller overheads in dataflow.

Keywords: Dataflow, Sparse matrix-vector multiplication, Trebuchet

Acknowledgments

We would like to thank FAPERJ and CAPES for the support given to the authors of this work.

---

1http://www.cise.ufl.edu/research/sparse/matrices/groups.html  
2http://software.intel.com/en-us/intel-mkl  
3http://openmp.org/
<table>
<thead>
<tr>
<th>Group</th>
<th>Matrix Name</th>
<th>Rows</th>
<th>Columns</th>
<th>Nonzeros</th>
<th>Data Type</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rudnyi</td>
<td>water tank</td>
<td>60,740</td>
<td>60,740</td>
<td>2,035,281</td>
<td>Real</td>
<td>Unsymmetric</td>
</tr>
<tr>
<td>Botonakis</td>
<td>FEM_3D_thermal2</td>
<td>147,900</td>
<td>147,900</td>
<td>3,489,300</td>
<td>Real</td>
<td>Unsymmetric</td>
</tr>
<tr>
<td>PARSEC</td>
<td>Ga41As41H72</td>
<td>268,096</td>
<td>268,096</td>
<td>18,488,476</td>
<td>Real</td>
<td>Symmetric</td>
</tr>
<tr>
<td>Freescale</td>
<td>circuit5M_dc</td>
<td>3,523,317</td>
<td>3,523,317</td>
<td>14,865,409</td>
<td>Real</td>
<td>Unsymmetric</td>
</tr>
</tbody>
</table>

Table 1: Matrices used for speedup tests.

![Speedup Results](image)

Figure 1: Speedup results for Intel MKL and Dataflow Trenbuchet implementations.

References


