Dynamic load balancing of the power method on a Grid

Clovis Dongmo Jiogo and Pierre Manneback

Faculté Polytechnique de Mons, Mons, Belgium
{Clovis.DongmoJiogo, Pierre.Manneback}@fpms.ac.be

1 Introduction

The power method is a simple iterative method based on the matrix vector product. This method is mostly used for calculating the eigenvector corresponding to the largest eigenvalue for large matrices. For example, it is used for calculating the page rank of web pages in the search engines.

An efficient execution of the power method on Grid environment requires matrix partitioning and distribution to take into account of the heterogeneity of resources. We propose in this paper a dynamic load balancing of the power method which performs, for each k iterations, an adjustment of the data distribution according to a factor. This factor is defined by the elapsed time required by the preceding iterations and aims to balance the computation load over the processors.

2 Dynamic distribution model

Several methods have been proposed for dynamic load balancing of sparse matrix vector product, which is the kernel computation of iterative methods such as the power method (e.g. [1, 2]): some of them try to overlap communication with computation; some others allow only the most overloaded processor to send rows to its neighbors. Another idea is to store matrix rows on several nodes redundantly and create overlapping regions.

We propose in this work, a deterministic iterative method which is based on a given predefined rules:

Let consider a set of heterogenous resources R_1,...,R_l. Each resource is composed of one or several processors. We denote p_i, the total amount of power available on the resource R_i.

Let $\Pi = \{A_1, A_2,...,A_k\}$ be a k-way row partition of sparse matrix A. For the initial load balancing, the matrix is partitioned according to resource power p_i representing the power factor. Each resource R_i receives the block A_i and the adjacent blocks A_{i-1} and A_{i+1}. After k iteration, we

make an adjustment by rebalancing the load. This adjustment of the load takes into account the preceding response time (computing time, communication time, latency) of processors and sets the new power factor in aim to minimize the total execution time. The general algorithm of the dynamic load balancing is stated as follows:

Algorithm :Iterative_Load_Balancing

```
s:=initialBalancing();
initialDistribution(s,A);
repeat{iterative improvement}
    computeIteration();
    distribution(s);
    r() := responseTime();
    after k iteration
        if unBalanced (r, ε)
        then s := adjustementBalancing(r);
until stopping_criteria
```

In this algorithm, the initial balancing splits the sparse matrix in block rows according to the resource power. The vector and the nonzero elements of the sparse matrix are then distributed to each resource. After k iterations of matrix vector product, if the load is unbalanced ($\epsilon \geq$ the standard deviation of elapsed time between processors), a adjustment is performed by examining the three factors: computing time, communication time and latency. Since each resource owns the adjacent blocks, this rebalancing do not involved a new redistribution of the sparse matrix. Only the vector with the new computing ranges are sent to the resources which continue the computations.

In the full paper, we will present some experimental results and parameter analysis for this dynamic load balancing strategy. Preliminary results have shown its interest.

References

1. Peter Christen, *A parallel iterative linear system solver with dynamic load balancing.* In Proceedings of the International Conference on Supercomputing ICS-98, July 13-17 1998, Melbourne, pages 7-12. ACM Press, 1998.
2. J. R. McCombs, R. T. Mills, and A. Stathopoulos, *Dynamic load balancing of an iterative eigensolver on networks of heterogeneous clusters.* In Proceedings of the 17th International Parallel and Distributed Processing Symposium, 2003.