QR Decomposition: Demonstration of Distributed Computing on Wireless Sensor Networks

Sherine Abdelhak, Soumik Ghosh, Rabi Chaudhuri, Magdy Bayoumi
The Center for Advanced Computer Studies, University of Louisiana at Lafayette
{spa9242, sxg5317, rsc5446, mab} @ cacs.louisiana.edu

Introduction

Wireless sensor nodes, despite their limited computational and power resources, can exhibit effective processing capabilities if collaborative distributed processing is explored. Distributed collaborative computing on wireless sensor nodes can reduce the energy consumption per node and speed up the response time of the system. The applications of wireless sensor networks in target tracking and environment monitoring pose a common computational problem that deals with signal processing tasks, like beamforming. An essential part of this computation relies on least squares solutions. QR decomposition is an efficient factorization that transforms a least squares problem into a triangular least squares problem which is easier to solve.

The QR decomposition of a matrix A of size $m \times n$ where $m \geq n$, gives the two matrices R and Q, where Q is an orthogonal matrix of size $m \times m$ and R is an upper triangular matrix of size $m \times n$ such that Eq.1 is satisfied.

$$A = Q \cdot R \quad (1)$$

The contributions of this work are developing a QR decomposition algorithm which is suitable for distribution and parallelization on wireless sensor networks, and implementing the distributed version of the proposed algorithm on a test bed of Telosb [2] wireless sensor nodes. Our experiments prove that the algorithm reduces the energy per node, on average, by 1.87x and speeds up the computation, on average, by 1.61x. This work lays the foundation for implementing other numerical methods for distributed processing over energy and resource-constrained wireless sensor nodes.

Proposed Algorithm

The proposed algorithm is inspired by the AllReduce method [3] and mixes Householder reflections and Givens rotation. Consider a $m \times n$ matrix A; the proposed algorithm has the following steps:

- Divide A into p vertical partitions $V_0, V_1, \cdots, V_p$. For simplicity we consider partitions of equal sizes $m \times v$.
- Each vertical partition V_j is further divided into q blocks $B_{0j}, B_{1j}, \cdots, B_{qj}$. The blocks are of equal sizes $b \times v$ where $b \geq v$.
- The algorithm is recursive where each iteration calculates the QR decomposition of *one* vertical partition. Thus the algorithm has p steps and the p^{th} step involves calculating the QR decomposition of only 1 block B_{qp}. The colored blocks in Figure 1a indicate those which are processed in each iteration for a matrix of size $9 \times 9$with $p = 3$, $q = 3$ and blocks of size 3×3. Each iteration involves the following steps:
- Calculate the QR decomposition, using Householder reflections, of the blocks $B_{ij}, \cdots, B_{qj}$ *simultaneously*, where j is the partition index and $i \geq j$. The calculation of the R matrix is done in-place, replacing the original block; while the Q matrix requires extra storage. The new partition $V_j^{'}$ has its blocks in upper triangular form.
- The blocks of $V_j^{'}$, except for the first block, will have their entries annihilated in this step using parallel Givens. The procedure is to couple two blocks B_x and B_y where B_x is called the *reference* block. B_y's entries are totally annihilated by using the rows of B_x. The annihilation using Givens rotations consists of v rotation matrices for a block size of $b \times v$. The sequence in which the elements of B_y are annihilated using the rows of B_x is illustrated in Figure 1b. It is very important to zero the entries in the specified order so that previously zeroed entries are not filled in again. At the end of this stage, $R_v$of $V_j^{'}$ is the resulting matrix, and Q_v is the product of all the Givens rotation matrices.

The Givens algorithm here is iterative and the number of iterations, g, depends on the number of blocks q (Eq.2).

$$g = log_2 q \quad (2)$$

- Using AllReduce, the QR decomposition of the original partition V_j is deduced from R_v and Q_v. AllReduce method is used for calculating the QR decomposition of long and "skinny" matrices where the data is partitioned horizontally, in a way similar to our scenario. Following AllReduce, Q_v is partitioned into QW_i partitions. The Q_i's calculated in the first step using householder reflections are now multiplied by the corresponding QW_i. The final Q is obtained by stacking the product $Q_i \times QW_i$ for each block. Figure 2 illustrates applying AllReduce on a partition V_j. Figure 3 illustrates the proposed algorithm for *p=3* and *q=3*. An important contribution of this work is using AllReduce in finding the final Q matrix from the vertical partitions of an original matrix; rather than calculating the R matrix only, as found in literature.

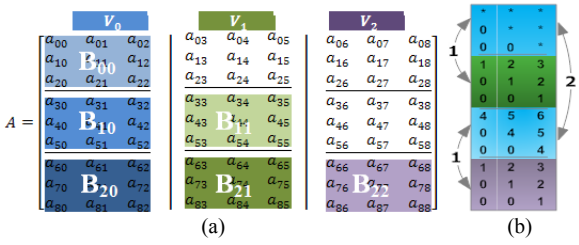

Figure 1: (a) Processed blocks for *p=3* and *q=3*, (b) Annihilation sequence in parallel Givens

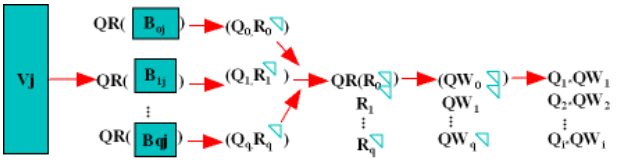

Figure 2: AllReduce method applied to a thin matrix V_j[3]

Proposed Distribution

The proposed distribution considers the scenario when one of the sensor nodes distributes the computation problem amongst its neighbors. In this case, the algorithm has the following phases:

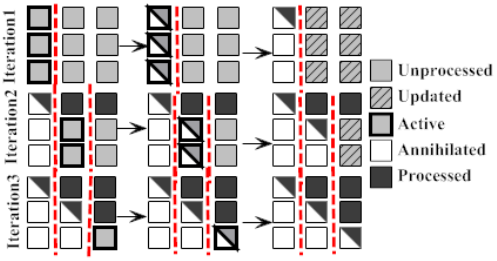

Figure 3: Illustration of the proposed algorithm

Initialization Phase: the distributing node broadcasts an INVITE message which specifies the computation operation and the size of the matrix to be distributed.

Sub-cluster Formation Phase: the neighboring nodes send STATUS messages to the distributing node and commit to joining the computation cluster. In the STATUS message, the nodes specify their residual voltage (RV) which is a physical layer parameter that indicates the battery level.

Task Mapping Phase: the distributing node determines the matrix partition parameters p and q, runs an energy-aware task allocation algorithm, and unicasts each task to the selected neighboring node. To determine q, the *simplest* and most *efficient* way is to divide each partition into q blocks where q is *equal* to the number of nodes to handle the computation. If q is *greater* than the number of nodes, then full parallelism in the QR calculation for each block, cannot be exploited. On the other hand, choosing a big q means that the partitions are smaller and more nodes are utilized; consequently, the communication overhead encountered to manage all the nodes involved can easily overcome the advantages of distributing the computation. The number of partitions, p, dictates the number of iterations of the algorithm. Therefore, smaller p (larger v) directly impacts the speed of the computation.

Considering that the distributing node has the same resource constraints as the other nodes in the WSN, it is necessary that the task allocation and scheduling algorithms are simple and light-weight. 'List' algorithm was used in which the distributing node maintains a list of the participating nodes and sorts it in decreasing order of RV. The first q nodes are selected as candidates. The blocks are unicasted to the corresponding nodes and each node is assigned a temporary code (ID) which is the index of the block assigned to it. The significance of this ID is explained in the next phase.

Computation and Result Gathering Phase: the computation is carried out by the nodes as described in the previous section, and the results are reported back to the distributing node. The main operations are:

- ***Householder reflection*** to obtain R and Q
- ***Parallel Givens*** where the nodes are coupled in pairs. One node (*reference* node) holds the reference block, and the other node (*annihilated* node) holds the block to be annihilated by the reference block. The *reference* node has an even ID, it sends its R to the *annihilated* node whose ID can be deduced according to Eq.3. The updated reference block is sent back to the *reference* node, and the G matrix calculated by the *annihilated* node is sent to the distributing node. The *distributing* node calculates Q_v as the product of all G matrices. At the end of this stage, the whole partition V_i is in upper triangular form. Therefore, the resulting matrix R_v is the R matrix of the original V_i, but the resulting Q_v is *not* the Q of V_i.

$$ID_{annihilated} = ID_{reference} + g \quad (3)$$

- ***AllReduce*** is carried out to obtain Q_{final} from Q_v. After calculating Q_v, the *distributing* node partitions it into blocks (QW_i) and unicasts the blocks to the corresponding nodes. The nodes simply multiply their local matrix Q_i, obtained in the first operation with QW_i. The result is reported to the *distributing* node which replaces the current partition by the new Q_{final}. The previous operations are repeated until all the partitions are processed. The final result is the multiplication of all Q_{final} matrices.

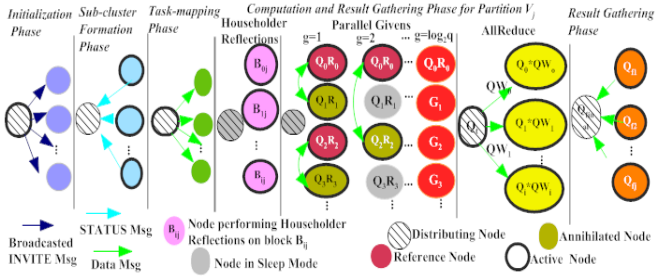

Figure 4: The proposed algorithm and its distribution

Experimental Results

The power consumption of the Telosb nodes was determined using the shunt resistor method [1]. Figure 5 shows the energy consumption *per node* and the time required for running the QR decomposition in centralized and distributed manner, for different matrix sizes. The proposed distribution achieves up to 2x speed up and up to 2.1x energy savings per node.

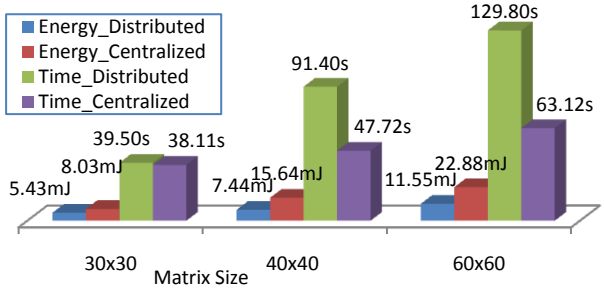

Figure 5: Energy *per node* (mJ) and time (sec) for Centralized and Distributed QR Decomposition for different matrix sizes

References

[1] A. Milenkovic, M. Milenkovic, E. Jovanov, D. Hite and D. Raskovic, "An environment for runtime power monitoring of wireless sensor network platforms", *37th Southeastern Symposium on System Theory*, Tuskegee, AL, Mar. 2005.

[2] J. Polastre, R. Szewczyk and D. Culler, Telos: "Enabling ultra-low power wireless research", *Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*, Apr 2005.

[3] J. Langou, "All Reduce Algorithms: Applications to QR factorization",*Tech. Report 80-02, ETH, Zurich, Switzerland*, 1980.