Resource-aware Distributed Block-based LU Decomposition on Wireless Sensor Networks

Sherine Abdelhak, Soumik Ghosh, Jared Tessier, Magdy Bayoumi
The Center for Advanced Computer Studies, University of Louisiana at Lafayette

Introduction

In this work, we describe the implementation of a distributed LU decomposition on wireless sensor networks. It forms part of a numerical methods *kernel* for distributed signal processing on wireless sensor networks.

Contribution

- Algorithm for distributing the LU decomposition of an NxN matrix
- Energy and storage-aware task allocation
- Fast and smooth recovery from node failure
- Efficient scheduling and time slot allocation
- Exhaustive analysis and experiments on Telosb platform

Background

LU decomposition of a 3x3 matrix involves finding lower and upper triangular matrices such that *L.U = A:*

$$\begin{bmatrix}\alpha_{11} & 0 & 0\\ \alpha_{21} & \alpha_{22} & 0\\ \alpha_{31} & \alpha_{32} & \alpha_{33}\end{bmatrix}\cdot\begin{bmatrix}\beta_{11} & \beta_{12} & \beta_{13}\\ 0 & \beta_{22} & \beta_{23}\\ 0 & 0 & \beta_{33}\end{bmatrix}=\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}$$

Crout's algorithm for finding the LU decomposition of a 3x3:

1) Set $\alpha_{ii} \equiv 1 \;\forall i = 1,\ldots,N$

2) solve for β_{ij} and α_{ij}

$$\beta_{ij} = a_{ij} - \sum_{k=1}^{i-1}\alpha_{ik}\beta_{kj} \qquad i = 1,2,\ldots,j$$

$$\forall j = 1,2,3,\ldots,N \qquad \alpha_{ij} = \frac{1}{\beta_{jj}}\left(a_{ij} - \sum_{k=1}^{j-1}\alpha_{ik}\beta_{kj}\right) \qquad i = j+1, j+2,\ldots,N$$

In *Block-based* LU decomposition, an *NxN* matrix A can be partitioned into at least 4 matrices: A_{11} is a *bxb* matrix, A_{12} is *bx(n-b)*, A_{21} is *(n-b)xb*, and A_{22} is *(n-b)x(n-b)*. *b* is the partition size.

$$A = \begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix} = \begin{bmatrix}L_{11} & 0\\ L_{21} & L_{22}\end{bmatrix}\cdot\begin{bmatrix}U_{11} & U_{12}\\ 0 & U_{22}\end{bmatrix}$$

References

[1] C. Nikias, A. Chrysafis, and A. Venetsanopoulos, "The LU decomposition theorem and its implications to the realization of two dimensional digital filters," IEEE transactions on acoustics, speech, and signal processing, vol. 33, no. 3, pp. 694–711, 1985.

[2] T. Hauser, A. Dasu, A. Sudarsanam, and S. Young, "Performance of a LU Decomposition on a Multi-FPGA System Compared to a Low Power Commodity Microprocessor System," Scalable Computing: Practice and Experience, vol. 8, no. 4, pp. 373–385.

[3] V. Daga, G. Govindu, V. Prasanna, S. Gangadharapalli, and V. Sridhar, "Efficient Floating-point based Block LU Decomposition on FPGAs," International Conference on Engineering of Reconfigurable Systems and Algorithms, pp. 21–24.

Acknowledgement

This work was supported in part by the U.S. Department of Energy under Award Number DE-FG02-04ER46136 and by the Louisiana Board of Regents under Contract Number DOE/LEQSF(2004-07)-ULL.

Proposed Work

a Computation

The algorithm is iterative, each *computational* iteration involves:

Stage1: Apply Crout's algorithm for finding the LU decomposition of A_{11}. Compute the 1st row of A_{12} knowing that α_{11} =1, similarly, compute the 1st column of A_{21} knowing that $\beta_{11} = a_{11}$.

Stage2: Compute A_{12} completely using L_{11} from Stage1 to obtain U_{12}; similarly, process A_{21} completely using U_{11} from Stage1 to obtain L_{21}. Compute the 1st term of each column of $L_{21}U_{12}$ for A_{new}.

Stage3: Compute A_{new} using the results from Stage2. If A_{new} is large enough to be partitioned again, Stage1 through 3 will be repeated; otherwise Crout's algorithm is directly applied to A_{new} to find its LU decomposition.

b Resource Allocation

1 partition in 1 message

$$nRounds = R = \begin{cases}\lfloor N/b \rfloor + 1 & ((N \bmod b) + b) > MAX\_MX\_SIZE\\ \lfloor N/b \rfloor & \text{otherwise}\end{cases}$$

$$nPartitions = P = R^2 \qquad nNodes = 2\times R - 1$$

Distinguish 4 kinds of sensor *nodes* involved in the computation in terms of the type of operations assigned to them:

- **ClusterHead** responsible for the LU operation using Crout's algorithm
- **Node with ID2** responsible for lower matrix operation. (even-ID nodes)
- **Grp1** responsible for upper matrix operation. (odd-ID nodes)
- **Grp2** responsible for partitioning/regrouping of the original matrix

Selection algorithm is resource-aware, in terms of power and storage. Three bins for voltage and storage are distinguished.

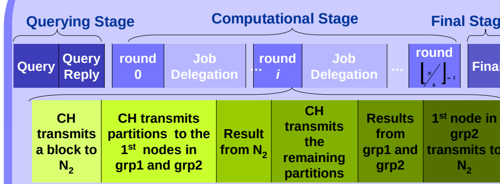
ENERGY and STORAGE requirements increase with the iteration number. Nodes in grp2 have higher requirements than those in grp1

- BS initially selects [N/b]² nodes out of which only (2*[N/b]-1) will be active, the rest can sleep and only wakeup during JobDelegation slots
- BS builds a CandidateList for each active node, out of the rest nodes
  - *CandidateList* contains (R-1) CandidateNodes. R = #rounds the node should be active.
  - *CandidateNodes* belong to the same voltage level and same memory level of the active node (within a variance ±δ)

Proposed Work

c Communication

Frame format:



1.Querying stage: BS broadcasts QUERY packet to query the voltage and storage levels of the nodes. BS also runs the selection algorithm.

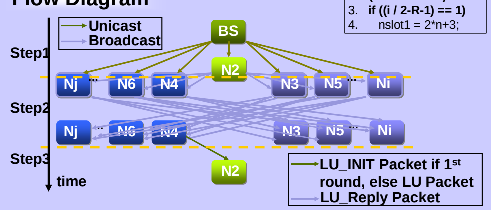
1.Computation Stage: formed of [N/b] rounds and [N/b] – 1 job delegation periods

2.Final Stage: BS regroups the result

Pseudo-code for each node to determine its time slot:

```
1.nslot = n + i -2 *R;
2.if (i mod 2 == 0)
3.   if ((i / 2-R-1) == 1)
4.      nslot1 = 2*n+3;
```

Flow Diagram



Failure Management by Auto-power Notification

Any node Ni wishes to retire:

- Sets RETIRE bit to 1
- Broadcasts LU_REPLY packet holding the result for the current operation during its assigned slot
- Broadcasts LU_REPLY packet holding the result of any extra calculation
- Delete the virtual address

BS gets LU_REPLY with RETIRE bit ON:

- Indexes the CandidateList with the address of the retired node
- Picks the 1st CandidateNode
- Sends JOB_DELEGATION packet to the new node containing
  - the same virtual address as the retired node (i)
  - the result of any extra calculation

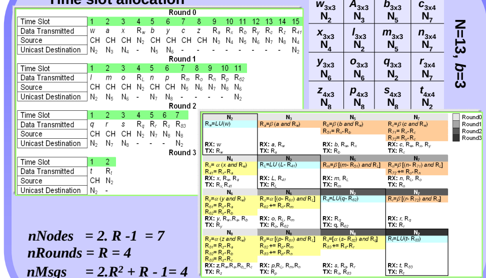
Overhead = 2 Msgs

New node wakes up during the JD slot, receives the JOB_DELEGATION and decodes its address from the message

- Adjusts its schedule based on the new address and stays awake

Illustrative Example

Time slot allocation



nNodes = 2. *R* -1 = 7
nRounds = *R* = 4
nMsgs = 2.R² + R - 1= 4

Detailed operation

Experiment

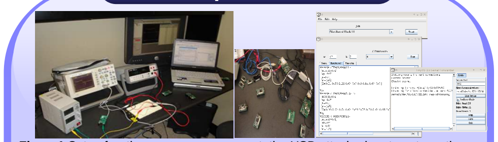


Figure1 Setup for the power measurement; the USB attached mote acts as the Cluster Head, the other motes are distributed around the laboratory. A java GUI was built to provide user friendly environment to deal with the motes.

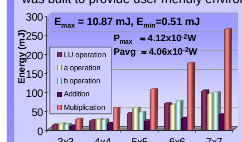


Figure 2 Total energy (mJ) for different matrix sizes and different partition sizes

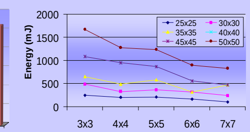
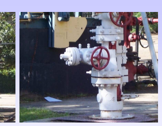


Figure 3 Total time (ms) per operation for different matrix sizes and different partition sizes

Applications

Sensors distributed on lifter and plunger, measuring vibrations and structural properties remotely in a distributed fashion. The distributed, embedded, numerical computational kernel, enables the processing tasks.



A practical application for vibration measurement on an oil derrick was done as part of project UCOMS (www.ucoms.org)