
Hybrid QR Factorization Algorithm for High Performance Computing Architectures

Peter Vouras

Naval Research Laboratory Radar Division

Professor G.G.L. Meyer

**Johns Hopkins University Parallel Computing and
Imaging Laboratory**

Report Documentation Page			Form Approved OMB No. 0704-0188			
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.						
1. REPORT DATE 21 MAY 2003		2. REPORT TYPE N/A		3. DATES COVERED -		
4. TITLE AND SUBTITLE Hybrid QR Factorization Algorithm for High Performance Computing Architectures				5a. CONTRACT NUMBER		
				5b. GRANT NUMBER		
				5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S)				5d. PROJECT NUMBER		
				5e. TASK NUMBER		
				5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) John Hopkins University Parallel Computing and Imaging Laboratory				8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)		
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release, distribution unlimited						
13. SUPPLEMENTARY NOTES The original document contains color images.						
14. ABSTRACT						
15. SUBJECT TERMS						
16. SECURITY CLASSIFICATION OF:				17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 23	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified				



- Background
- Problem Statement
- Givens Task
- Householder Task
- Paths Through Dependency Graph
- Parameterized Algorithms
- Parameters Used
- Results
- Conclusion



- In many least squares problems, QR decomposition is employed
 - Factor matrix A into unitary matrix Q and upper triangular matrix R such that $A = QR$
- Two primary algorithms available to compute QR decomposition
 - Givens rotations
 - ◆ Pre-multiplying rows $i-1$ and i of a matrix A by a 2x2 Givens rotation matrix will zero the entry $A(i, j)$
 - Householder reflections
 - ◆ When a column of A is multiplied by an appropriate Householder reflection, it is possible to zero all the subdiagonal entries in that column

$$\begin{bmatrix} * & * & * & * \\ 0 & * & * & * \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} * & * & * & * \\ * & * & * & * \end{bmatrix}$$

$$\begin{bmatrix} * & * & * & * \\ 0 & * & * & * \end{bmatrix} = \left(I - \frac{2}{v^T v} v v^T \right) \begin{bmatrix} * & * & * & * \\ * & * & * & * \end{bmatrix}$$



- Want to minimize the latency incurred when computing the QR decomposition of a matrix A and maintain performance across different platforms
- Algorithm consists of parallel Givens task and serial Householder task
- Parallel Givens task
 - Allocate blocks of rows to different processors. Each processor uses Givens rotations to zero all available entries within block such that
 - ◆ $A(i, j) = 0$ only if $A(i-1, j-1) = 0$ and $A(i, j-1) = 0$
- Serial Householder task
 - Once Givens task terminates, all distributed rows are sent to root processor which utilizes Householder reflections to zero remaining entries



*	*	*	*	*	*	*	*	*	*
0	*	*	*	*	*	*	*	*	*
0	0	*	*	*	*	*	*	*	*
0	0	0	*	*	*	*	*	*	*
Processor 0									
*	*	*	*	*	*	*	*	*	*
0	*	*	*	*	*	*	*	*	*
0	0	*	*	*	*	*	*	*	*
Processor 1									
*	*	*	*	*	*	*	*	*	*
0	*	*	*	*	*	*	*	*	*
0	0	*	*	*	*	*	*	*	*
Processor 2									
*	*	*	*	*	*	*	*	*	*
0	*	*	*	*	*	*	*	*	*
0	0	*	*	*	*	*	*	*	*

- Each processor uses Givens rotations to zero entries up to the topmost row in the assigned group
- Once task is complete, rows are returned to the root processor
- Givens rotations are accumulated in a separate matrix before updating all of the columns in the array
 - Avoids updating columns that will not be use by an immediately following Givens rotation
 - Saves significant fraction of computational flops



Processor 0	*	*	*	*	*	*	*	*	*	*
0	*	*	*	*	*	*	*	*	*	*
0	0	*	*	*	*	*	*	*	*	*
0	0	0	*	*	*	*	*	*	*	*
0	0	0	*	*	*	*	*	*	*	*
0	0	0	*	*	*	*	*	*	*	*
0	0	0	*	*	*	*	*	*	*	*
0	0	0	*	*	*	*	*	*	*	*
0	0	0	*	*	*	*	*	*	*	*
0	0	0	*	*	*	*	*	*	*	*

- Root processor utilizes Householder reflections to zero remaining entries in Givens columns
- By computing a-priori where zeroes will be after each Givens task is complete, root processor can perform a sparse matrix multiply when performing a Householder update for additional speed-up
 - Householder update is $A = A - \beta vv^T A$
- Householder update involves matrix-vector multiplication and an outer product update
 - Makes extensive use of BLAS routines



*	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
8	17	*	*	*	*	*	*	*	*
7	16	24	*	*	*	*	*	*	*
6	15	23	30	*	*	*	*	*	*
5	14	22	29	35	*	*	*	*	*
4	13	21	28	34	39	*	*	*	*
3	12	20	27	33	38	42	*	*	*
2	11	19	26	32	37	41	44	*	*
1	10	18	25	31	36	40	43	45	*

- Algorithm must zero matrix entries in such an order that previously zeroed entries are not filled-in
- Implies that $A(i, j)$ can be zeroed only if $A(i-1, j-1)$ and $A(i, j-1)$ are already zero
- More than one sequence exists to zero entries such that above constraint is satisfied
- Choice of path through dependency graph greatly affects performance

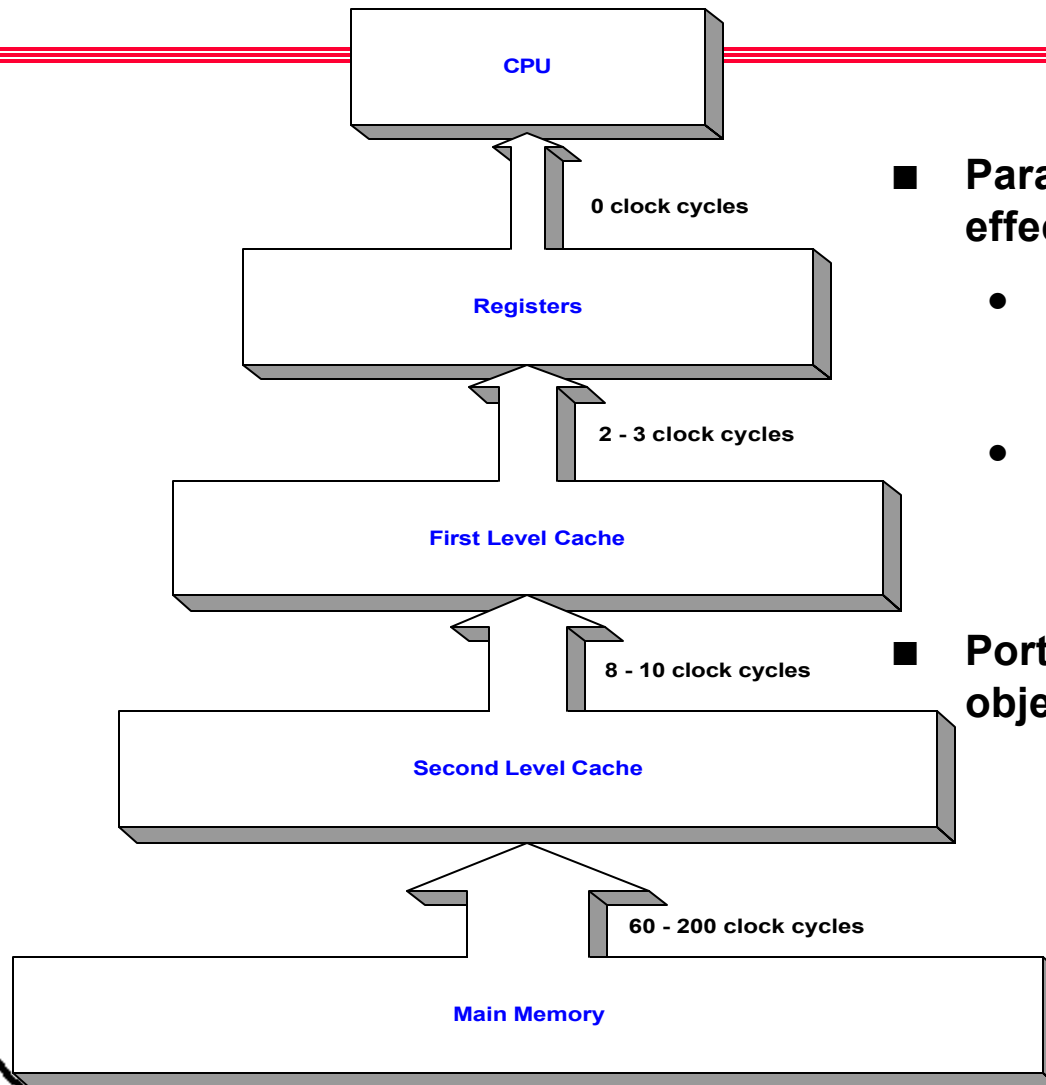


*	*	*	*	*	*	*	*	*	*
37	*	*	*	*	*	*	*	*	*
29	38	*	*	*	*	*	*	*	*
22	30	39	*	*	*	*	*	*	*
16	23	31	40	*	*	*	*	*	*
11	17	24	32	41	*	*	*	*	*
7	12	18	25	33	42	*	*	*	*
4	8	13	19	26	34	43	*	*	*
2	5	9	14	20	27	35	44	*	*
1	3	6	10	15	21	28	36	45	*

■ By traversing dependency graph in zig-zag fashion, cache line reuse is maximized

- Data from row already in cache is used to zero several matrix entries before row is expunged from cache

Parameterized Algorithms : Memory Hierarchy

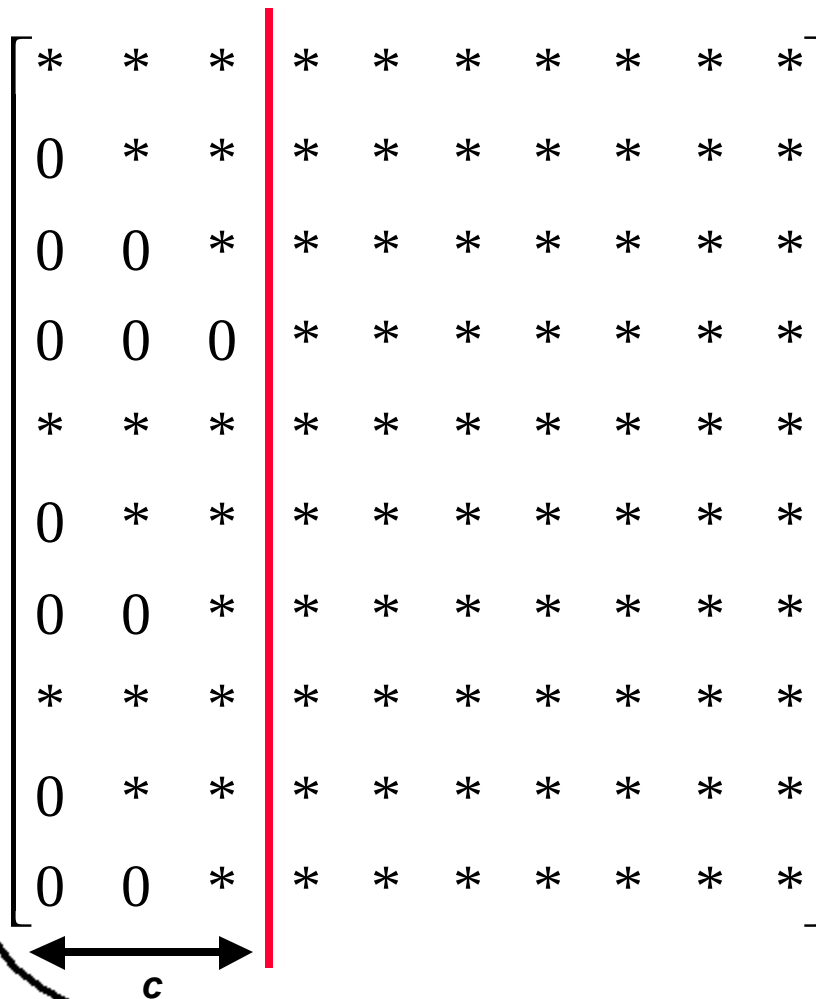


Memory Hierarchy of SGI O2000

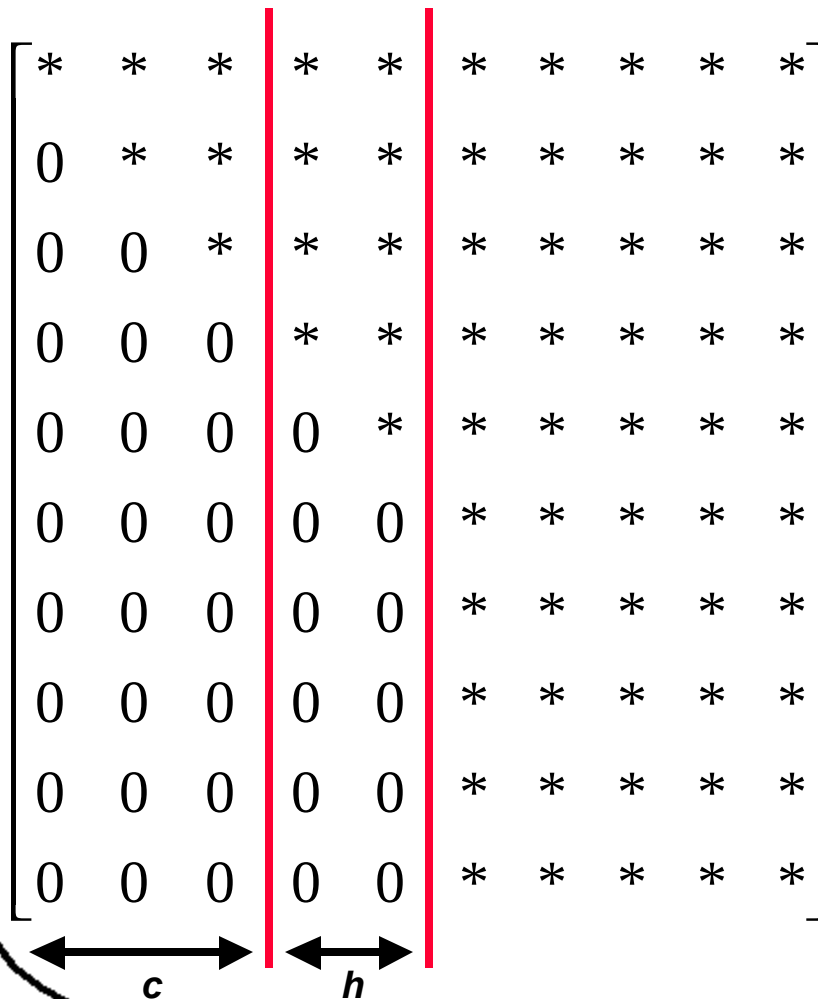
■ **Parameterized Algorithms make effective use of memory hierarchy**

- Improve spatial locality of memory references by grouping together data used at the same time
- Improve temporal locality of memory references by using data retrieved from cache as many times as possible before cache is flushed

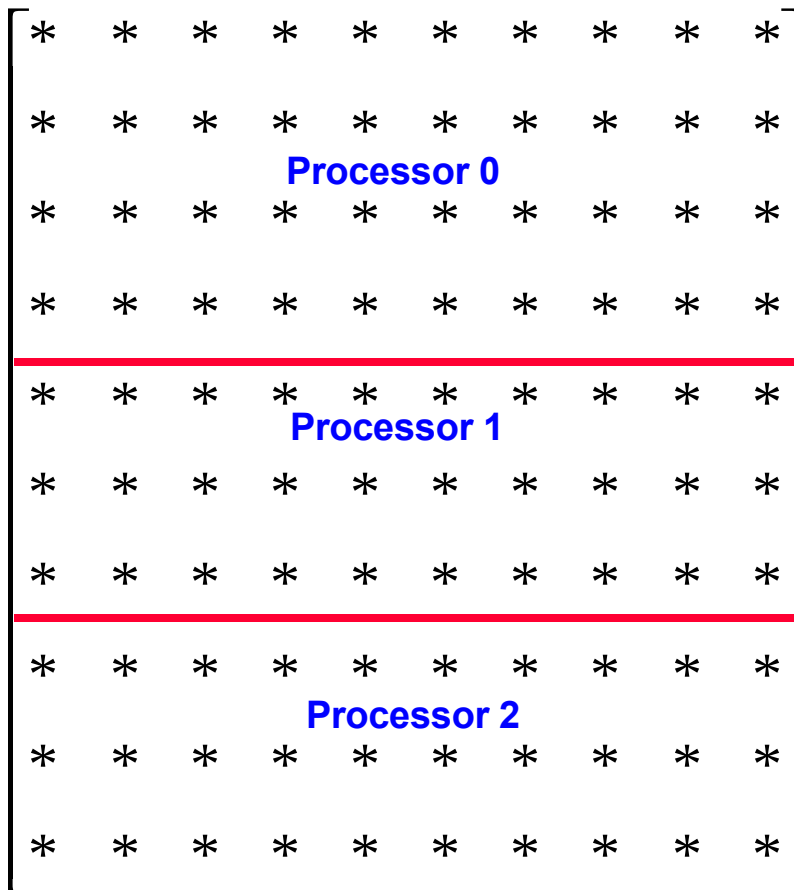
■ **Portable performance is primary objective**



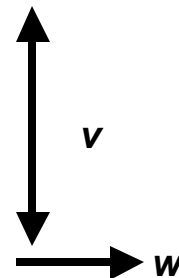
- Parameter c controls the number of columns in Givens task
- Determines how many matrix entries can be zeroed before rows are flushed from cache



- Parameter h controls the number of columns zeroed by Householder reflections at the root processor
- If h is large, the root processor performs more serial work, avoiding the communication costs associated with the Givens task
- However, the other processors sit idle longer, decreasing the efficiency of the algorithm



- Parameters v and w allow operator to assign rows to processors such that the work load is balanced and processor idle time is minimized



Results



- 48 550-MHz PA-RISC 8600 CPUs
- 1.5 MB on-chip cache per CPU
- 1 GB RAM / Processor

HP Superdome

SPAWAR in San Diego, CA



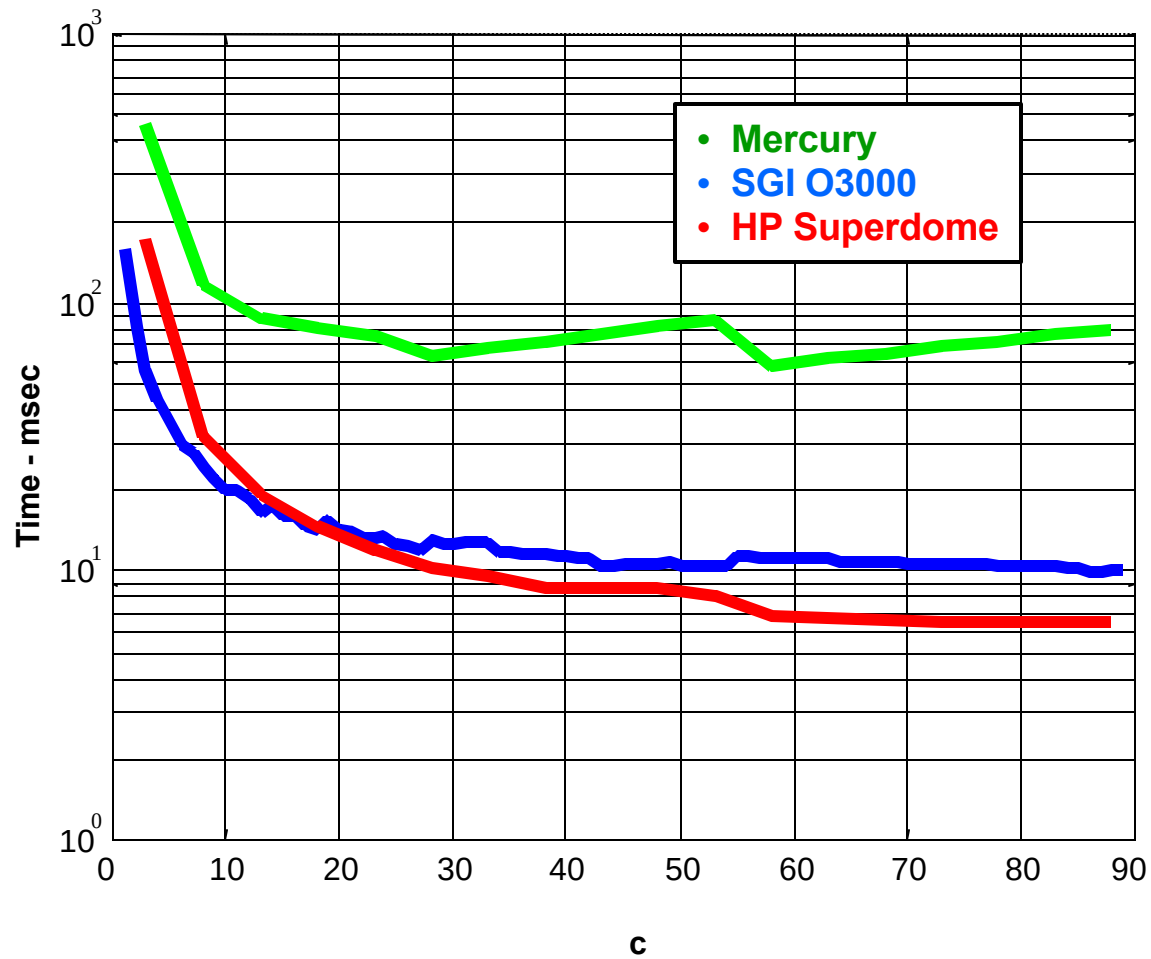
- 512 R12000 processors running at 400 MHz
- 8 MB on-chip cache
- Up to 2 GB RAM / Processor

SGI O3000**NRL in Washington, D.C.**

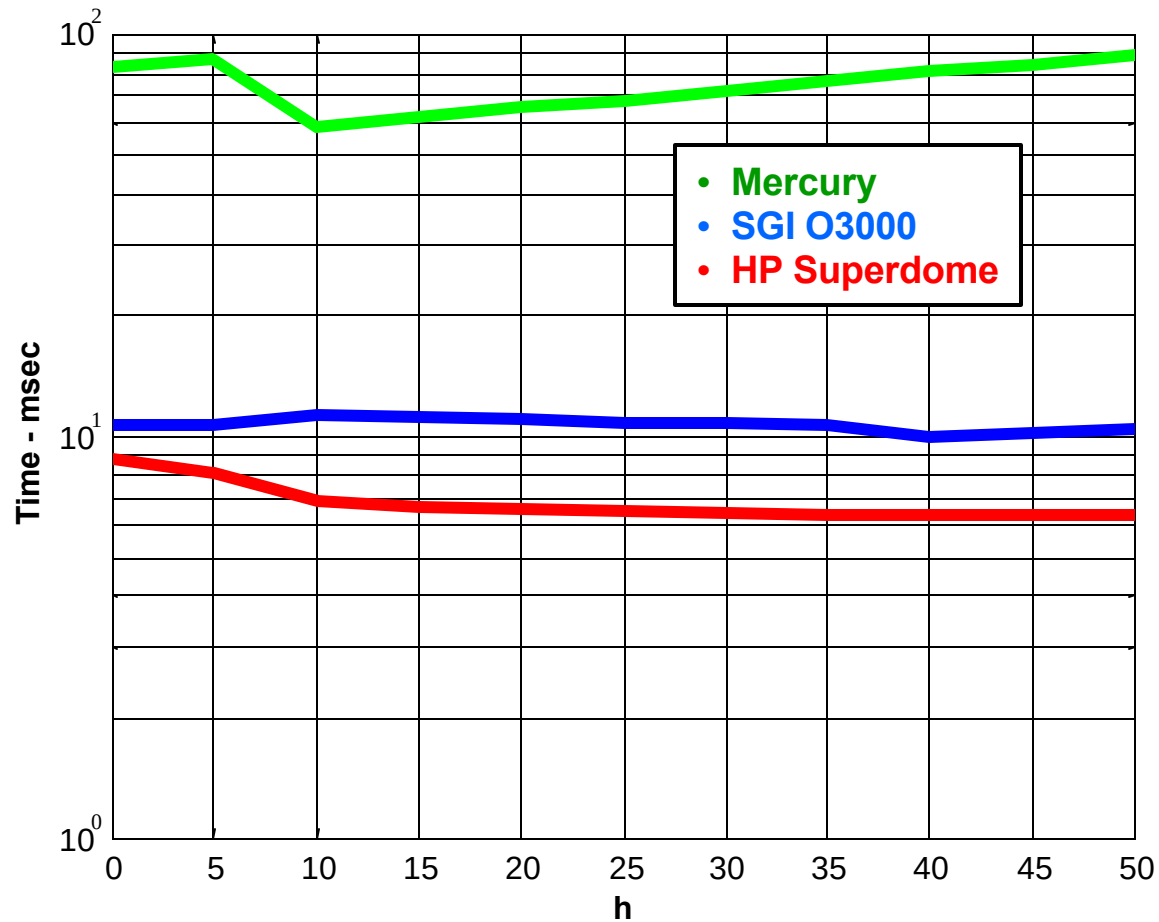


- 8 Motorola 7400 processors with AltiVec units
- 400 MHz clock
- 64 MB RAM per processor

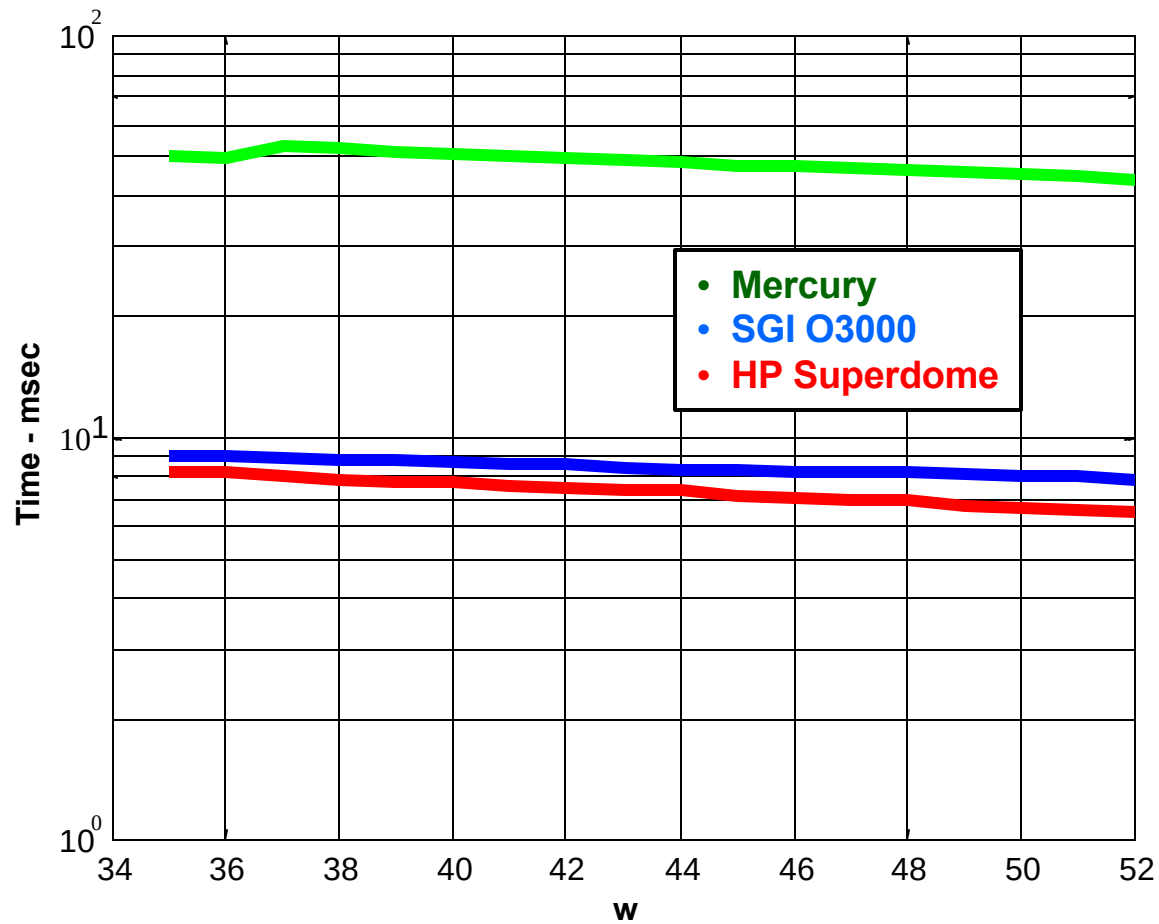
Mercury**JHU in Baltimore, MD**



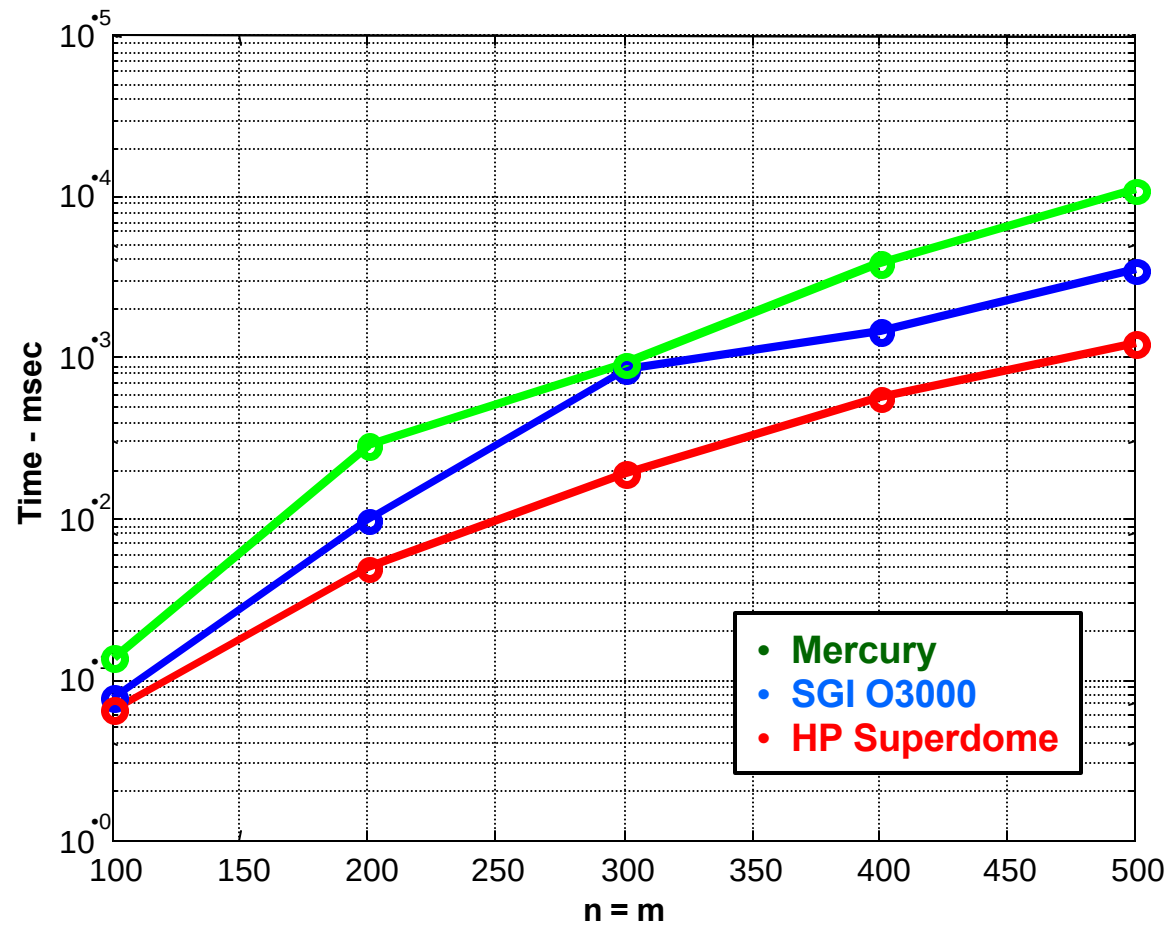
100 x 100 array
4 processors
 $p = 12, h = 0$

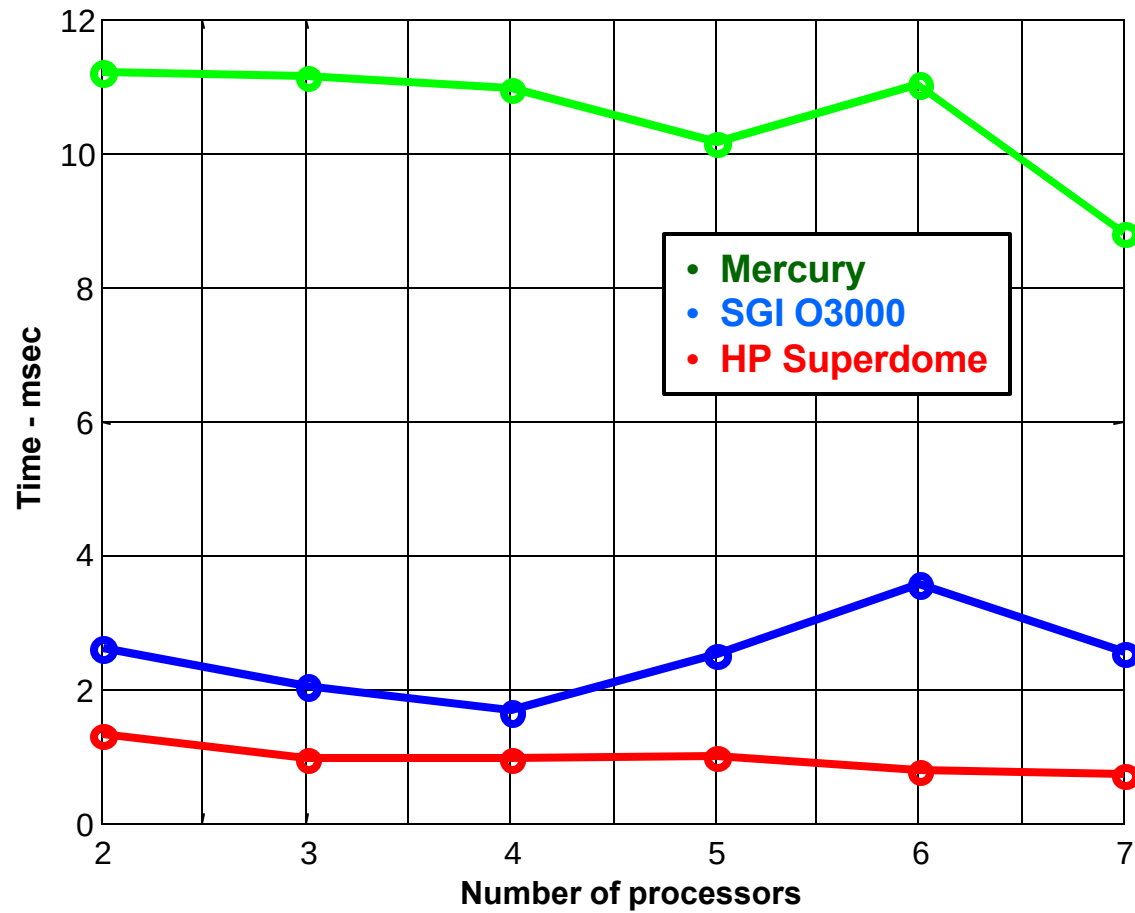


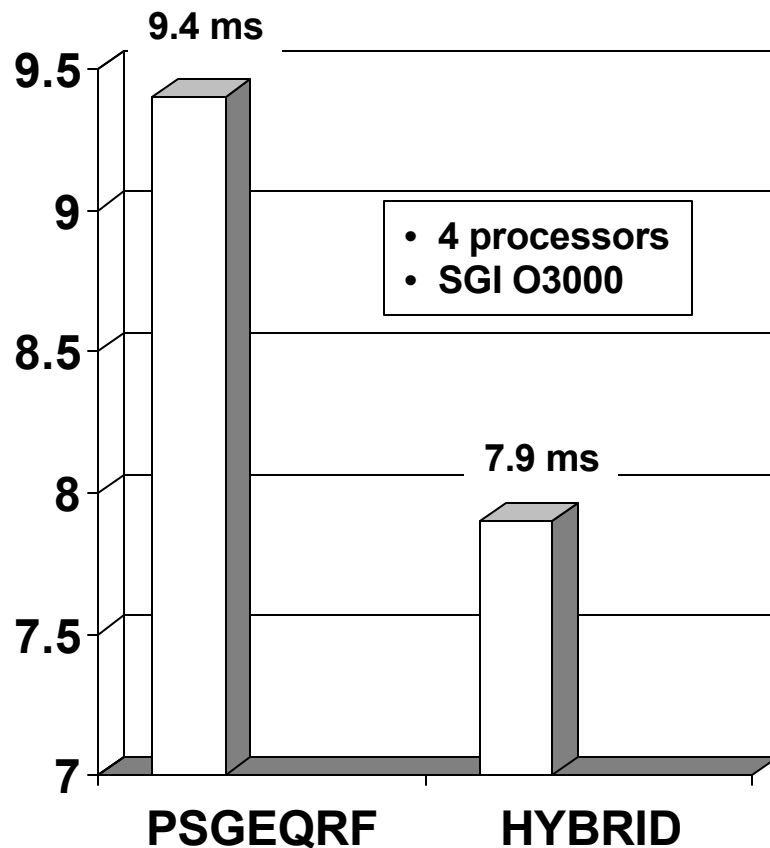
100 x 100 array
4 processors
 $c = 63, p = 12$



100 x 100 array
4 processors
 $h = 15, p = 10,$
 $c = 60, v = 15$







- For matrix sizes on the order of 100 by 100, the Hybrid QR algorithm outperforms the SCALAPACK library routine PSGEQRF by 16%
- Data distributed in block cyclic fashion before executing PSGEQRF



- Hybrid QR algorithm using combination of Givens rotations and Householder reflections is efficient way to compute QR decomposition for small arrays on the order of 100×100
- Algorithm implemented on SGI O3000 and HP Superdome servers as well as Mercury G4 embedded computer
- Mercury implementation lacked optimized BLAS routines and as a consequence performance was significantly slower
- Algorithm has applications to signal processing problems such as adaptive nulling where strict latency targets must be satisfied