The LINPACK Benchmark on the Fujitsu AP 1000

Richard P. Brent

Computer Sciences Laboratory

Australian National University

The LINPACK Benchmark

A popular benchmark for floating-point performance.

Involves the solution of a nonsingular system of *n* equations in *n* unknowns by Gaussian elimination with partial pivoting.

Three Cases

n = 100

The original benchmark (too easy for our purposes).

n = 1000

Often used to compare vector processors and parallel computers.

n >> 1000

Often used to compare massively parallel computers.

Assumptions

Assume double-precision arithmetic (64-bit).

Interested in $n \geq 1000$.

Assume coefficient matrix available in processors.

Use C indexing conventions -

- **Indices 0, 1, ...**
- **Row-major ordering**

Hardware

The *Fujitsu AP 1000* (also known as the *CAP II*) is a MIMD machine with up to 1024 independent 25 Mhz Sparc processors (called *cells*).

Each cell has 16 MB RAM, 128 KB cache, and Weitek floating-point unit capable of 5.56 Mflop for overlapped multiply and add.

Communication

The topology of the AP1000 is a torus with wormhole routing. The theoretical bandwidth between any pair of cells is 25 MB/sec.

In practice, because of system overheads, copying of buffers, etc, only about 6 MB/sec is attainable by user programs.

Data Distribution

Possible ways of storing matrices (data and results) on the AP 1000 are -

- **column wrapped**
- **row wrapped**
- **scattered =**
  row and column wrapped
- **blocked versions of these**

We chose the *scattered* representation because of its good load-balancing and communication bandwidth properties.

Scattered Storage

On a 2 by 2 configuration

cell cell
cell cell

a 4 by 6 matrix would be stored as follows, where the color-coding indicates the cell where an element is stored -

00	01	02	03	04	05
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35

Scattered Storage - Global ↔ Local Mapping

On a machine configuration with *ncelx . ncely* cells *(x, y)*, $0 \leq x < ncelx, 0 \leq y < ncely$, element $a_{i,j}$ is stored in cell

$$(j \bmod ncelx,\ i \bmod ncely)$$

with local indices[1]

$$i' = i \text{ div } ncely,$$
$$j' = j \text{ div } ncelx.$$

---

[1] **Sorry about the confusing *(i,j)* and *(y,x)* conventions !**

Blocked Storage

If the above definition of scattered storage is applied to a block matrix with *b* by *b* blocks, then we get the *blocked panel-wrapped* representation. Choosing larger *b* reduces the number of communication steps but worsens the load balance.

We use $b = 1$, but $b > 1$ has been used on other local-memory machines (e.g. Intel Delta).

Blocked Matrix Operations

The rank-1 updates in Gaussian elimination can be grouped into blocks of ω so rank-ω updates can be performed using level 3 BLAS (i.e. matrix-matrix operations).

The two possible forms of blocking are independent - we can have $b > 1$ or $\omega > 1$ or both. If both then $b = \omega$ is convenient but not necessary. In our implementation

$$b = 1, \quad \omega \geq 1.$$

Gaussian Elimination

The idea of Gaussian Elimination (G.E.) is to transform a nonsingular linear system

$$Ax = b$$

into an equivalent upper triangular system

$$Ux = b'$$

which is (relatively) easy to solve for x. It is also called LU Factorization because

$$PA = LU,$$

where P is a permutation matrix and L is lower triangular.

A Typical Step of G.E.

```
x x x x x x x
  x x x x x x
    x x x x x
    x x x x x
    x x x x x
    x x x x x
```

is converted by row operations (rank-1 update) into

```
x x x x x x x
  x x x x x x
    x x x x x
    0 x'x'x'x'
    0 x'x'x'x'
    0 x'x'x'x'
```

Comments

x is a nonzero element,
x is the pivot element,
x is an element to be zeroed,
x is in the pivot row,
x → x' is in the active region.

Row interchanges are generally necessary to bring the pivot element x into the correct position.

The right-hand side vector has been stored as the last column of the (augmented) matrix.

Communication Requirements for G.E.

Pivot selection requires finding the largest element in (part of) a column; then, if necessary, two rows are interchanged. (We do this explicitly.)

The rank-1 update requires vertical broadcast (y_brd) of the pivot row and horizontal broadcast (x_brd) of the multiplier column.

x_brd and y_brd

The AP 1000 has hardware support for x_brd and y_brd, so these can be performed in the same time as a single cell to cell communication.
(A binary tree with *O*(log *n*) communication overheads is not required.)

x x x x x y_brd
↓ ↓ ↓ ↓ ↓

x →
x → x_brd
x →

Memory Refs per Flop

The ratio

$$R = (\text{loads and stores})/(\text{flops})$$

is important because it is impossible to keep the floating-point unit busy unless $R < 1$. Rank-1 updates

$$a_{ij} \leftarrow a_{ij} + u_i * v_j$$

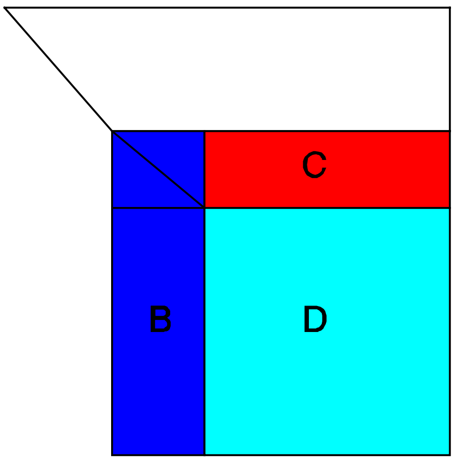
have $R \geq 1$. To reduce R and improve performance, need blocking. (ω rank-1 updates $\rightarrow$ one rank-ω update.)

G.E. with Blocking

Defer operations on the region labelled D until ω steps of G.E. have been performed. Then the rank-ω update is simply

$$D \leftarrow D - BC$$

and can be performed by level-3 BLAS without inter-cell communication.



Choice of ω

Operations in the vertical strip of width ω and the horizontal strip of depth ω are done using rank-1 updates (slow) so want ω to be small. However, level-3 BLAS for rank-ω updates are slow unless ω is large. The optimum choice is usually

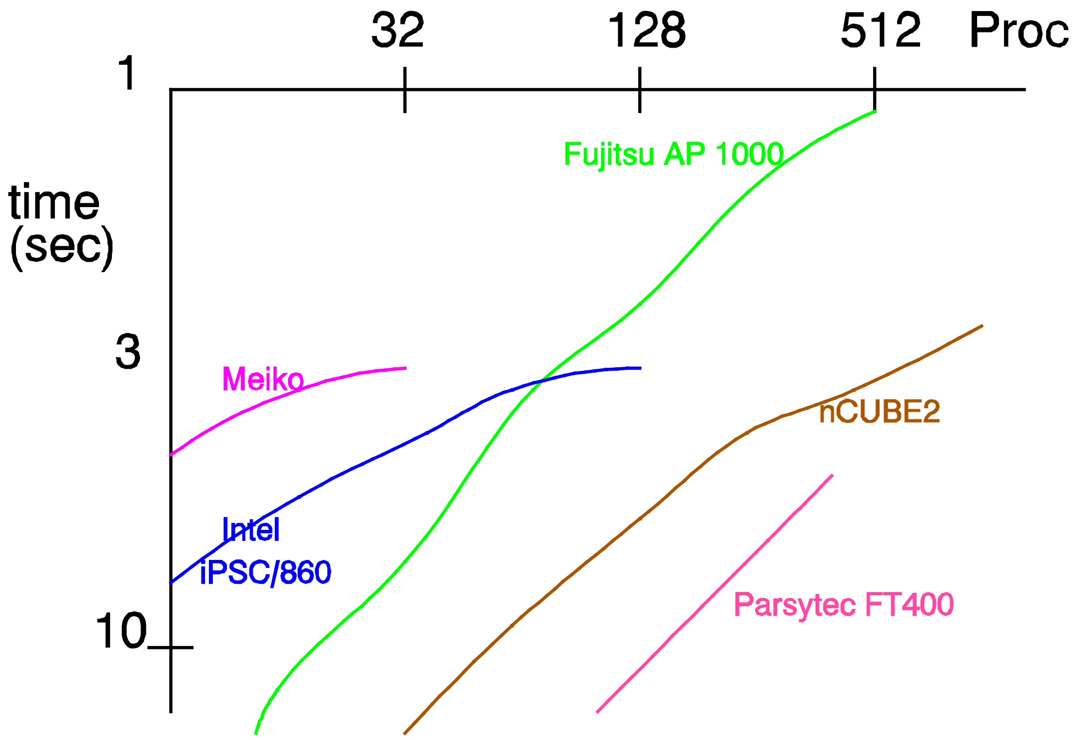
$$\omega \sim n^{1/2}$$

However, ω should be small enough that the parts of the strips stored on each cell fit in the cache.

LINPACK Benchmark Results *(n = 1000)* on the AP 1000

cells	time (sec)	speedup	efficiency
512	1.10	147	0.29
256	1.50	108	0.42
128	2.42	66.5	0.52
64	3.51	46.0	0.72
32	6.71	24.0	0.75
16	11.5	13.9	0.87
8	22.6	7.12	0.89
4	41.3	3.90	0.97
2	81.4	1.98	0.99
1	160	1.00	1.00

Comparison for *n = 1000* using Dongarra's Table 2



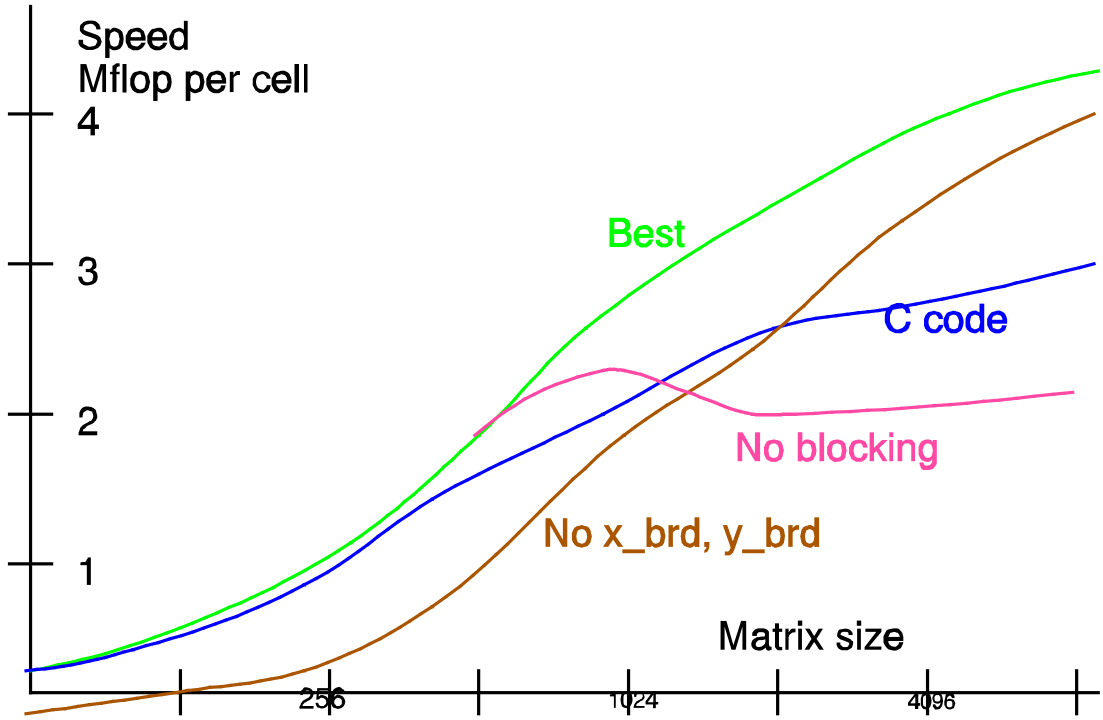
The AP 1000 is fastest for ≥ 128 cells and has little "tailoff" as number of cells ↑

LINPACK Benchmark Results *(n large)* on the AP 1000

cells	r_{max} Gflop	n_{max}	n_{half}	r_{max}/r_{peak}
512	2.251	25600	2500	0.79
256	1.162	18000	1600	0.82
128	0.566	12800	1100	0.80
64	0.291	10000	648	0.82
32	0.143	7000	520	0.80
16	0.073	5000	320	0.82

Note the high ratio r_{max}/r_{peak} and the large ratio n_{max}/n_{half}.

Comparison of Options on 64-cell AP 1000



The graph shows the effect of turning off blocking, hardware x_brd, y_brd, or assembler BLAS 3 inner loops.

Conclusions

The Fujitsu AP 1000 is a well-balanced machine for linear algebra. It is possible to attain at least 50% of peak performance over a wide range of problem sizes.

Hardware support for x and y broadcast is a good feature.

The communication speed is high and startup costs low relative to the floating-point speed (which is slow by current standards).