12 January 2015

Jakub Ciecierski

An upper triangular dense matrix is split by columns on N processors. Implement the backward substitution. Analyze the speed-up.

Parallel Processing Project 4

1 Algorithm description

Algorithm 1 Sequential Backwards substitution

Data: Ax=b
for *i=n-1 to 1* **do**
    $x[i] = b[i]/A[i,i]$
    for *j=0 to i-1* **do**
        $b[j] = b[j] - x[i] * A[j,i]$
    end
end

At i-th iteration, the process which has i-th column, computes x_i and updates the b vector. When worker is done computing, he sends the next worker updated vectors b and x. Since each i-th iteration requires data from the previous iteration, column-wise parallel backward substitution is scarcely a parallel algorithm and actually works faster sequentially as I will show in the next section.

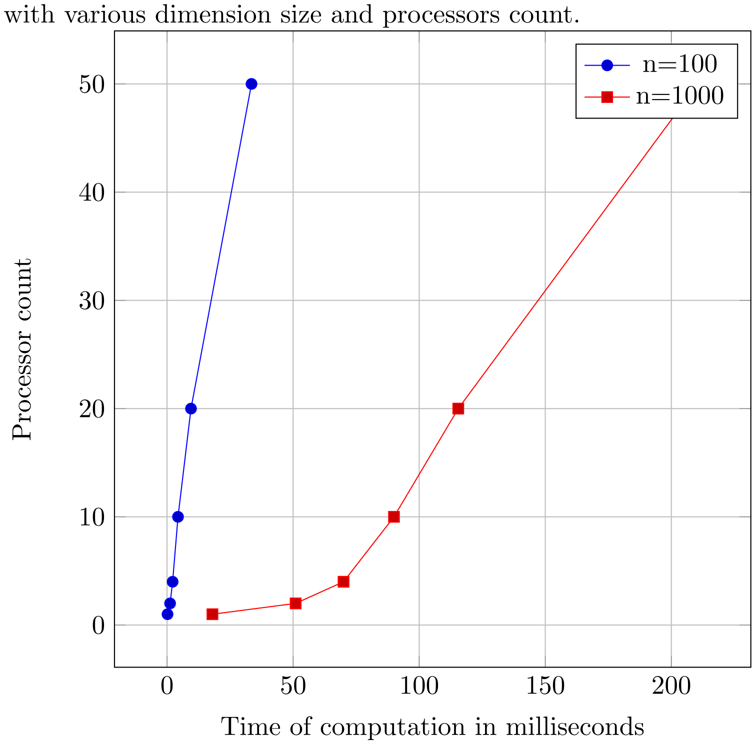
2 Analysis of speed up

I created upper triangular dense matrices using a simple formula:

Algorithm 2 Column major order Matrix

```
Data: n = Dimension of A; val = 1; c = r = 0
while c > n do
    while r > n do
        if r > c then
            A[c * n + r] = 0
        else
            A[c * n + r] = val
            val + +
        end
    end
end
```

with various dimension size and processors count.



The plot clearly shows that the algorithm is not really parallel. In fact the computations are the fastest with one process.