INF236 PARALLEL ALGORITHMS - ASSIGNMENT 2 2014

General rules

- The assignment is due April 1 at 4 p.m.
- Your answer is to be uploaded at the course page at MiSide.
- Your answer should be a zip-file containing program files and a brief report on how the problem was solved (see specifications below).
- Grading system: 0–100 points. Passing the assignment with at least 40 points is a prerequisite for admission to the exam. The total score on all three compulsory assignments counts 50% of the final grade.

Problem

Given the function $f_n(x) = -3 + \sum_{i=1}^{n}\sum_{j=1}^{n} \sin\frac{x^i+x^j}{2}\left(1-\cos\frac{x^i+x^j}{2}\right)$ where $n > 2$ is some integer, we want to find the unique root $r \in [0,1]$ such that $f_n(r) = 0$. Uniqueness of the root is easily verified by observing that $f_n(0) < 0$, $f_n(1) > 0$, and that f_n is continuous and monotonously increasing on the interval $[0,1]$. In this project, you are supposed to write MPI-programs that, for given values of $\varepsilon > 0$, compute an interval $[\ell, u]$ of width no more than ε such that $r \in [\ell, u]$.

a) Assume we divide the interval $[0,1]$ into p subintervals $[0,\frac{1}{p}], [\frac{1}{p},\frac{2}{p}], \ldots, [\frac{p-1}{p},1]$. To check whether $r \in [\frac{q}{p},\frac{q+1}{p}]$, we can let process q compute and check the sign of $f_n\left(\frac{q}{p}\right)$ and $f_n\left(\frac{q+1}{p}\right)$. If the signs of f_n differ at the two endpoints of the interval, we know that $r \in [\frac{q}{p},\frac{q+1}{p}]$, otherwise $r \notin [\frac{q}{p},\frac{q+1}{p}]$. Thus, exactly one of the processes will conclude positively, and the corresponding interval can be subdivided into p subintervals, whereas intervals corresponding to processes concluding negatively will not be considered further. This process is continued until the width of the subintervals is no more than ε.

   Write an MPI program following this idea. Once all processes have concluded the investigation of their corresponding interval, they exchange information such that they all know in which interval r is located (a master-slave approach is not to be followed here). Derive an analytical expression for the speedup as a function of n, ε and p. Let $\varepsilon = 2^{-30}$ and $n = 100$, and make experiments with $p = 2$, $p = 4$, $p = 8$ and $p = 32$. For each experiment, measure the speedup by comparing to a sequential version based on binary search. Next, repeat the experiments with $n = 1000$.

b) The approach indicated in question a) can immediately be improved. Observe that processes $q-1$ and q both compute $f_n\left(\frac{q}{p}\right)$. Also, (the sign of) f_n is known at the endpoints of the interval, and there is for instance no need to compute $f_n(0)$ and $f_n(1)$. Alternatively, we can select the p

evenly disributed points $\frac{1}{p+1}, \frac{2}{p+1}, \ldots, \frac{p}{p+1}$, and let process q compute only $f_n\left(\frac{q+1}{p+1}\right)$ $(q = 0, \ldots, p-1)$. After these computations, the processes jointly hold sufficient information to conclude in which of the intervals $\left[0, \frac{1}{p+1}\right], \left[\frac{1}{p+1}, \frac{2}{p+1}\right], \ldots, \left[\frac{p}{p+1}, 1\right]$ r is located. Next, the same idea is applied recursively to the interval in question until the interval widths drop below ε.

Write an MPI program for this approach, and do the experiments suggested in question a). Also, compare the running times of the two approaches, and comment briefly.

c) Finally, you should try to improve the performance further by developing a master-slave solution. Symmetry gives $f_n(x) = -3 +$

$$\sum_{i=1}^{n} \left(\sin x^i \left(1 - \cos x^i\right) + 2 \sum_{j=i+1}^{n} \sin \frac{x^i + x^j}{2} \left(1 - \cos \frac{x^i + x^j}{2}\right) \right) \quad (1)$$

Since the terms in the sums are non-increasing with increasing indices i and j, and since each term is non-negative, we have that $f_n(x) \in [L_{nk}(x), U_{nk}(x)]$, where $L_{nk}(x) = -3 +$

$$\sum_{i=1}^{k} \left(\sin x^i \left(1 - \cos x^i\right) + 2 \sum_{j=i+1}^{n} \sin \frac{x^i + x^j}{2} \left(1 - \cos \frac{x^i + x^j}{2}\right) \right) \quad (2)$$

and $U_{nk}(x) = L_{nk}(x) + (n-k)^2 \sin x^k \left(1 - \cos x^k\right)$. The lower bound $L_{nk}(x)$ is derived by letting the outer sum in (1) run only to k, and the upper bound $U_{nk}(x)$ is obtained by replacing all $(n-k)^2$ terms hence omitted by the larger value $\sin x^k \left(1 - \cos x^k\right)$. It follows that if $L_{nk}(x) \geq 0$ then $f_n(x) \geq 0$, and if $U_{nk}(x) \leq 0$ then $f_n(x) \leq 0$. Consequently, a process evaluating the sign of $f_n(x)$ can conclude after having run only k outer iterations of (1) whenever $L_{nk}(x) \geq 0$ or $U_{nk}(x) \leq 0$.

In a master-slave algorithm, the slaves evaluate $f_n(x)$ at carefully chosen points x. Once a slave concludes about the sign, it notifies the master, which assigns a new evaluation point to the slave. Observe that if a process concludes that $f_n(x) \geq 0$, the conclusion from a process evaluating the sign of $f_n(y)$ is irrelevant if $y \geq x$. Analogously, if $f_n(x) \leq 0$, the answer from a process evaluating $f_n(y)$ is not interesting if $y \leq x$. Therefore, we might let the master *interrupt* any slave if the master discovers that the current task of the slave is no longer relevant. This can be accomplished by letting the slaves ask the master regularly whether they should continue or discontinue their current task. The *frequency* of such requests is an important design parameter. For instance, a request can be sent after each computation of a new term in the sum (1). Assessing the frequency is a matter of trade-off between saving communication overhead and avoiding superfluous computations.

Develop and implement a master-slave approach using MPI for the problem in question. Run it on instances where n lies in the range from 100 to 1000 and p varies between 2 and 32, and try to achieve better performance

than what you got in questions a) and b). Results should be documented in tables or clear graphical illustrations. If you do not use $L_{nk}(x)$ and $U_{nk}(x)$, your method for fast conclusion about the sign of $f_n(x)$ should be described precisely and concisely.

Specifications

- The parameters ε and n are to be input to the program, preferably from the command line.
- Write a *concise* report where you describe how you solved the problem. Brief, concise reports with explicit conclusions rank higher than lengthy and verbose reports.