CENG 478

Assignment 4
Deadline: Jun 3, 23:59

Full Name: Yasin Fatih ALPUL
Id Number: 2098739

Answer 1

We can see that the processor that we should find moves in a circular way. That is, the P_{i+1} processor starts from process 0, continues with $1, 2, 3$ etc. and all the way up to the processor P_{p-1} and then go back to 0. Therefore, the $V(p)$ function is equal to $V(p) = p$, which is defined as after every $V(p)$ requests, each processor has received at least one work request.

The processor in this "edge" can be found in $\Theta(logp)$ time using the algorithm presented in (Lin, 1992). The algorithm is basically a binary search. The exact algorithm can be found below.

```
step=Proc_num; /* number of processors */
base=Myself; /*my processor Id */

val=remote_read(base);
while (step > 1)
{
    step = (step+1)/2;
    curr = (base+step)%Proc_num;

    temp = remote_read(curr);
    if (curr > base)
    {
        if (val == temp)
        {
            base = curr;
        }
    }
    else
    {
        if (val != temp)
        {
            base = curr;
            val = temp;
        }
    }
}
if ((base+1)%Proc_num == 0) /* round trip */
```

```
{
    val++;
}

next = (base+1)%Proc_num;
test_and_set(next, val); /* atomic operation */
```

If we assume this process is computed for p processes in each step and if we also assume there are a total of $logW$ steps, (let W be the total work, it gets divided in every step), then the total communication overhead becomes $O(log(p)log(W)p)$.

Answer 2

When the algorithm first started, all of the open lists of processors are initially empty. In this period, some nodes may be needlessly pass between processors. However, after some period of time, the algorithm will start to discard "bad" nodes.

Let some node n be a bad node and would not be expanded in a sequential algorithm. Since the hash function is completely random, that is, it maps each processor with equal $1/p$ probability, on average, the nodes which are expanded from that node will be discarded by other processors. Hence, the expected number of nodes expanded by this parallel formulation differs from the optimal number by a constant factor.

According to section 5.4.5 from the textbook, isoefficiency can be defined as the maximum of communication, concurrency and other overheads. Since in each iteration of the algorithm, $\Theta(p)$ nodes are expanded, and the cost of communicating one node is given as $O(1)$, it corresponds to a communication overhead of $\Theta(p)$. Since there are no other overheads, the isoefficiency function can be given as $\Theta(p)$.

Answer 3

a.

Speedup is defined as $\frac{T_s}{T_p}$ where T_s is the sequential execution time and T_p is the parallel execution time. Using the values given,

$$\begin{aligned}
\frac{T_s}{T_p} &= \frac{t_c n log n}{\frac{t_c n log n}{p} + \frac{t_w n log p}{p}} \\
&= \frac{1}{\frac{1}{p} + \frac{t_w n log p}{p t_c n log n}} \\
&= \frac{p}{1 + \frac{t_w n log p}{t_c n log n}} \\
&= \frac{p}{1 + \frac{t_w log p}{t_c log n}}
\end{aligned}$$

Efficiency is defined as speedup divided by number of processors. Using the speedup we have calculated

above,

$$\frac{\text{Speedup}}{p} = \frac{\frac{p}{1+\frac{t_w logp}{t_c logn}}}{p}$$
$$= \frac{1}{1+\frac{t_w logp}{t_c logn}}$$

If we use the given t_w and t_c values, we get

$$S = \frac{p}{1+0.2\frac{logp}{logn}}$$
$$E = \frac{1}{1+0.2\frac{logp}{logn}}$$

b.

According to the textbook, *isoefficiency* is defined as the growth rate of W required to keep the efficiency fixed as p increases. Using algebraic operations, we use the expression found above,

$$E = \frac{1}{1+0.2\frac{logp}{logn}}$$
$$\frac{1}{E} = 1+0.2\frac{logp}{logn}$$
$$\frac{1}{E} - 1 = 0.2\frac{logp}{logn}$$
$$\frac{5}{E} - 5 = \frac{logp}{logn}$$
$$logn = \frac{logp}{\frac{5}{E} - 5}$$
$$logn = \frac{Elogp}{5-5E}$$
$$logn = logp^{\frac{5}{5-5E}}$$
$$n = p^{\frac{5}{5-5E}}$$

We need to find the expression for W in order to find the isoefficiency function. Since W is defined in terms of n and the *Fast Fourier Transform* algorithm has $W = \Theta(nlogn)$, we find this expression by multiplying the last two equations side by side.

$$nlogn = p^{\frac{5}{5-5E}} logp^{\frac{5}{5-5E}}$$

Since the desired isoefficiency is 0.6, we use $E = 0.6$.

$$nlogn = 2.5p^{2.5}logp$$

c.

As the previous section, if we use $E = 0.4$, we get

$$nlogn = \frac{5}{3}p^{\frac{5}{3}}logp$$

d.

Using the expression we found in previous sections,

$$S = \frac{p}{1 + \frac{t_w logp}{t_c logn}}$$
$$E = \frac{1}{1 + \frac{t_w logp}{t_c logn}}$$

Using the given t_w and t_c values we get

$$S = \frac{p}{1 + \frac{logp}{logn}}$$
$$E = \frac{1}{1 + \frac{logp}{logn}}$$

If we compute the isoefficiency function as before, we get

$$E = \frac{1}{1 + \frac{logp}{logn}}$$
$$\frac{1}{E} = 1 + \frac{logp}{logn}$$
$$\frac{1}{E} - 1 = \frac{logp}{logn}$$
$$logn = \frac{logp}{\frac{1}{E} - 1}$$
$$logn = \frac{Elogp}{1 - E}$$
$$n = p^{\frac{E}{1-E}}$$
$$nlogn = p^{\frac{E}{1-E}} \frac{Elogp}{1 - E}$$

For $E = 0.6$ we get

$$nlogn = \frac{3}{2}p^{\frac{3}{2}}logp$$