CS 525 Homework 3

Junchao Yan
yan114@purdue.edu

February 26, 2014

1 Implement CG

1. *Create a Table in which you list the number of processes, the residual norm of the solution computed by your method, and the run times for the two experiments.*

   On one machine On distinct machine Difference

Process	Time	Speedup	Residual norm (L_2)
1	13.6159939766	1	9.928748e-6
2	7.3196909428	1.8601870056	9.90864e-6
4	4.318198204	3.1531655874	9.945577e-6
8	3.5294539928	3.8578188027	9.944483e-6

Process	Time	Speedup	Residual norm (L_2)
1	13.5593571663	1	9.928748e-6
2	8.5884821415	1.5787838809	9.90864e-6
4	6.876488924	1.9718430897	9.945577e-6
8	6.7402479649	2.0117000497	9.944483e-6

2. *Explain the speed-ups you observe for each experiment, and also the differences in speed-up between the two experiments.*

   For the experiment on one machine, the result shows that as the number of threads increased, the speedup also increased. Meanwhile, the experiment on distinct machines also had the same pattern. However, the difference is that the speedups for experiment on distinct machines are less than those on one machine. This is because the cost time for communicating between different machines is much more than that for communicating between processes in one machine.

2 Time Complexity of CG

By calculating the time complexity for each step, we can get the total time complexity.

1. Sparse matrix multiplication: As we assumed the number of nonzeros are equally distributed, $2(\frac{nnz(A)}{n}+1)*\frac{n}{nproc} = \frac{2}{nproc}(nnz(a)+n)$
2. Calculate $< r_k, r_k >$ and $< p_k, q_k >$ locally: $\frac{4n}{nproc}$
3. All reduce $< r_k, r_k >$: all to all broadcast $(nproc-1)*(t_{start}+t_{work})$
4. All reduce $< p_k, q_k >$: all to all broadcast $(nproc-1)*(t_{start}+t_{work})$
5. Calculate α: **1**
6. Update x and r_{k+1} locally: $\frac{4n}{nproc}$
7. Calculate $< r_{k+1}, r_{k+1} >$ locally: $\frac{2n}{nproc}$
8. All reduce $< r_{k+1}, r_{k+1} >$: all to all broadcast $(nproc-1)*(t_{start}+t_{work})$
9. Calculate β: **1**
10. Update p_{k+1} locally: $\frac{2n}{nproc}$
11. All gather p_{k+1}: all to all broadcast $(nproc-1)*(t_{start}+\frac{n}{nproc}*t_{work})$

Total: $\frac{2}{nproc}(nnz(A)+6n)+2+(nproc-1)(4t_{start}+(3+n/nproc)t_{work})$

Therefore, the worst case is $O(\frac{1}{nproc}(nnz(A)+n+nt_{work})+nproc*t_{start})$.

3 Modified version of the parallel CG algorithm

1. **Modified algorithm**

```
Initialization: r, p, q, x, local_q

while (iter < MAX_ITER){

// Sparse matrix multiplication
        for (i = 0; i < local_n; i++){
                tmp = 0;
                k1 = rbegin[i];
                k2 = rbegin[i+1]-1;
                if (k2 < k1){
                        continue;
                }
                for (k = k1; k <= k2; k++){
```

```
                        j = colind[k];
                        tmp += value[k]*p[j];
                }
                local_q[i] = tmp;
        }

MPI_Allgather(local_q,local_n,MPI_DOUBLE,q,local_n,MPI_DOUBLE,MPI_COMM_WORLD);

// Update alpha
        sum_r = 0;
        sum_pq = 0;
        for (i = 0; i < n; i++){
                sum_r += pow(r[i],2);
                sum_pq += p[i]*q[i];
          }

        alpha = sum_r/sum_pq;

// Update x and r
        for (i = 0; i < n; i++){
                x[i]+=alpha*p[i];
                r_next[i] = r[i] - alpha*q[i];
                r[i] = r_next[i];
        }

// Update residual
        sum_r_next = 0;
        for (i = 0; i < n; i++){
                sum_r_next += pow(r_next[i],2);
        }

// Check condition
        if (sqrt(sum_r_next) < CONST_TOL) break;

// Update beta
        beta = sum_r_next/sum_r;

        for (i = 0; i < n; i++){
                p[i] = r[i]+beta*p[i];
        }

} // end while

return x;
```

2. **Time complexity**

Similarly, the time complexity can be calculated step by step.

(a) Sparse matrix multiplication: As we assumed the number of nonzeros are equally distributed, $2(\frac{nnz(A)}{n} + 1) * \frac{n}{nproc} = \frac{2}{nproc}(nnz(a) + n)$

(b) All gather q_k: all to all broadcast $(nproc - 1) * (t_{start} + \frac{n}{nproc} * t_{work})$

(c) Calculate $< r_k, r_k >$ and $< p_k, q_k >$: $4n$

(d) Calculate α: **1**

(e) Update x and r_{k+1}: $4n$

(f) Calculate $< r_{k+1}, r_{k+1} >$: $2n$

(g) Calculate β: **1**

(h) Update p_{k+1}: $2n$

Total: $\frac{2}{nproc}(nnz(a) + n) + 12n + 2 + (nproc - 1) * (t_{start} + \frac{n}{nproc} * t_{work})$

Therefore, the worst case is $O(\frac{1}{nproc}(nnz(A) + n + nt_{work}) + n + nproc * t_{start})$.