TMA4280
Exercise 4

Vegard Stenhjem Hagen

February 18, 2013

Serial

```c
int main(int argc, char** argv){
    double pi = 4.0*atan(1);
    double sum = pi*pi/6;
    double time_init;

    if(argc < 2) {
        printf("Need one parameter, the size of the vector\n");
        return 1;
    }
    long int n = atoi(argv[1]);
    printf("Serial:\n");
    double* v = (double*)malloc(n*sizeof(double));
    double sumn = 0;
    time_init = walltime();

    for(long int i=n; i>0; i--){
        v[i] = 1.0/((double)i*i);
        sumn += v[i];
    }
    printf("Error:\t\t%e \nTime Elapsed:\t%f\n",sum-sumn,walltime()-time_init);
        return 0;
}
```

Parallel

OpenMP

```c
int main(int argc, char** argv){
    double pi = 4.0*atan(1);
    double sum = pi*pi/6;
    double time_init;

    if(argc < 2) {
        printf("Need one parameter, the size of the vector\n");
        return 1;
    }
    long int n = atoi(argv[1]);
    printf("OpenMP\tThreadcount: %i\n",omp_get_max_threads());
    double* v = (double*)malloc(n*sizeof(double));
    double sumn = 0;
    time_init = walltime();

#pragma omp parallel for schedule(static) reduction(+:sumn)
    for(long int i=n; i>0; i--){
        v[i] = 1.0/((double)i*i);
        sumn += v[i];
    }
    printf("Error:\t\t%e \nTime Elapsed:\t%f\n",sum-sumn,walltime()-time_init);
    return 0;
}
```

MPI

```c
int main(int argc, char** argv){
    int rank = 0,size = 1;
    double pi = 4.0*atan(1), sum = pi*pi/6;
    double time_init;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    if(rank == 0){
        printf("MPI   \tThreadcount: %i\n",size);
        if(argc < 2) {
            printf("Need one parameter, the size of the vector\n");
            MPI_Finalize();
            return 1;
        }else if(!isPowerOfTwo(size)){
            printf("The number of processors must be a power of 2");
            MPI_Finalize();
            return 1;
        }
    }
    int N = atoi(argv[1]), share = N/size;
    double sumn = 0.0;
    double* v = (double*)calloc(share,sizeof(double));
    time_init = walltime();

    for(int i=share; i>0; i--){
        v[i-1] = 1.0/(((double)i+rank*share)*(i+rank*share));
        sumn += v[i-1];
    }
    double s2 = sumn;
    MPI_Reduce(&s2, &sumn, 1, MPI_DOUBLE, MPI_SUM, 0,MPI_COMM_WORLD);

    if(rank == 0){
        printf("Error:\t\t%e \nTime Elapsed:\t%f \n",sum-sumn,walltime()-time_init);
    }
    MPI_Finalize();
    return 0;
}
```