SEQUENTIAL Parallel Lab

**void** work\_it\_seq(**long** \*old, **long** \*new) {

**int** i, j, k;

**int** u, v, w;

**long** compute\_it;

**long** aggregate=1.0;

**for** (i=1; i<DIM-1; i++) {

**for** (j=1; j<DIM-1; j++) {

**for** (k=1; k<DIM-1; k++) {

compute\_it = old[i\*DIM\*DIM+j\*DIM+k] \* we\_need\_the\_func();

aggregate+= compute\_it / gimmie\_the\_func();

}

}

}

printf("AGGR:%ld\n",aggregate);

**for** (i=1; i<DIM-1; i++) {

**for** (j=1; j<DIM-1; j++) {

**for** (k=1; k<DIM-1; k++) {

new[i\*DIM\*DIM+j\*DIM+k]=0;

**for** (u=-1; u<=1; u++) {

**for** (v=-1; v<=1; v++) {

**for** (w=-1; w<=1; w++) {

new[i\*DIM\*DIM+j\*DIM+k]+=old[(i+u)\*DIM\*DIM+(j+v)\*DIM+(k+w)];

}

}

}

new[i\*DIM\*DIM+j\*DIM+k]/=27;

}

}}

**for** (i=1; i<DIM-1; i++) {

**for** (j=1; j<DIM-1; j++) {

**for** (k=1; k<DIM-1; k++) {

u=(new[i\*DIM\*DIM+j\*DIM+k]/100);

**if** (u<=0) u=0;

**if** (u>=9) u=9;

histogrammy[u]++;

}

}

}

}

FAST Parallel Lab

**void** work\_it\_par(**long** \*old, **long** \*new) {

**const** **int** DIM\_SQUARED = DIM \* DIM;

**const** **int** TILE\_SIZE = 4;

**int** memory\_address, temp\_memory\_address, i, j, k, ii, jj, kk = 0;

**long** index0, index1, index2, index3, index4, index5 ,index6, index7, index8, index9, u, temp\_sum, compute\_it = 0;

**long** aggregate = 1.0;

#pragma omp parallel for private(j, k, ii, jj, kk, u, compute\_it, memory\_address, temp\_memory\_address) reduction(+: aggregate) reduction(+:temp\_sum) reduction(+:index0) reduction(+:index1) reduction(+:index2) reduction(+:index3) reduction(+:index4) reduction(+:index5) reduction(+:index6) reduction(+:index7) reduction(+:index8) reduction(+:index9)

**for** (i=1; i<DIM-1; i+= TILE\_SIZE) {

**for** (j=1; j<DIM-1; j+= TILE\_SIZE) {

**for** (k=1; k<DIM-1; k+= TILE\_SIZE) {

**for**(ii = i; (ii < i + TILE\_SIZE && ii < DIM - 1); ii ++) {

**for**(jj = j; (jj < j + TILE\_SIZE && jj < DIM - 1); jj ++) {

**for**(kk = k; (kk < k + TILE\_SIZE && kk < DIM - 1); kk ++) {

memory\_address = ii \* DIM\_SQUARED + jj \* DIM + kk;

compute\_it = old[memory\_address] \* we\_need\_the\_func();

aggregate += compute\_it / gimmie\_the\_func();

temp\_sum = 0;

temp\_memory\_address = memory\_address - DIM\_SQUARED;

temp\_memory\_address -= DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_memory\_address += DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_memory\_address += DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_memory\_address = memory\_address;

temp\_memory\_address -= DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_memory\_address += DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_memory\_address += DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_memory\_address = memory\_address + DIM\_SQUARED;

temp\_memory\_address -= DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_memory\_address += DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_memory\_address += DIM;

temp\_sum += old[temp\_memory\_address-1];

temp\_sum += old[temp\_memory\_address];

temp\_sum += old[temp\_memory\_address+1];

temp\_sum /= 27;

new[memory\_address] = temp\_sum;

u = temp\_sum / 100;

**if** (u <= 0) index0++;

**else** **if**(u == 1) index1++;

**else** **if**(u == 2) index2++;

**else** **if**(u == 3) index3++;

**else** **if**(u == 4) index4++;

**else** **if**(u == 5) index5++;

**else** **if**(u == 6) index6++;

**else** **if**(u == 7) index7++;

**else** **if**(u == 8) index8++;

**else** **if**(u >= 9) index9++;

}}}

}}}

printf("AGGR:%ld\n",aggregate);

histogrammy[0] = index0;

histogrammy[1] = index1;

histogrammy[2] = index2;

histogrammy[3] = index3;

histogrammy[4] = index4;

histogrammy[5] = index5;

histogrammy[6] = index6;

histogrammy[7] = index7;

histogrammy[8] = index8;

histogrammy[9] = index9;

}