Assignment 7 – Task 1

Dense triangular MVM

```
#pragma omp parallel private(i,k)
{
  for(k=0; k<niter; k++){
#pragma omp for schedule(runtime)
    for(j=0; j<size; ++j)
      for(i=0; i<=j; ++i)
        c[j]=c[j]+a[i+size*j]*b[i];
    if(c[size >> 1]<0.0) whatever();
  }
}
```

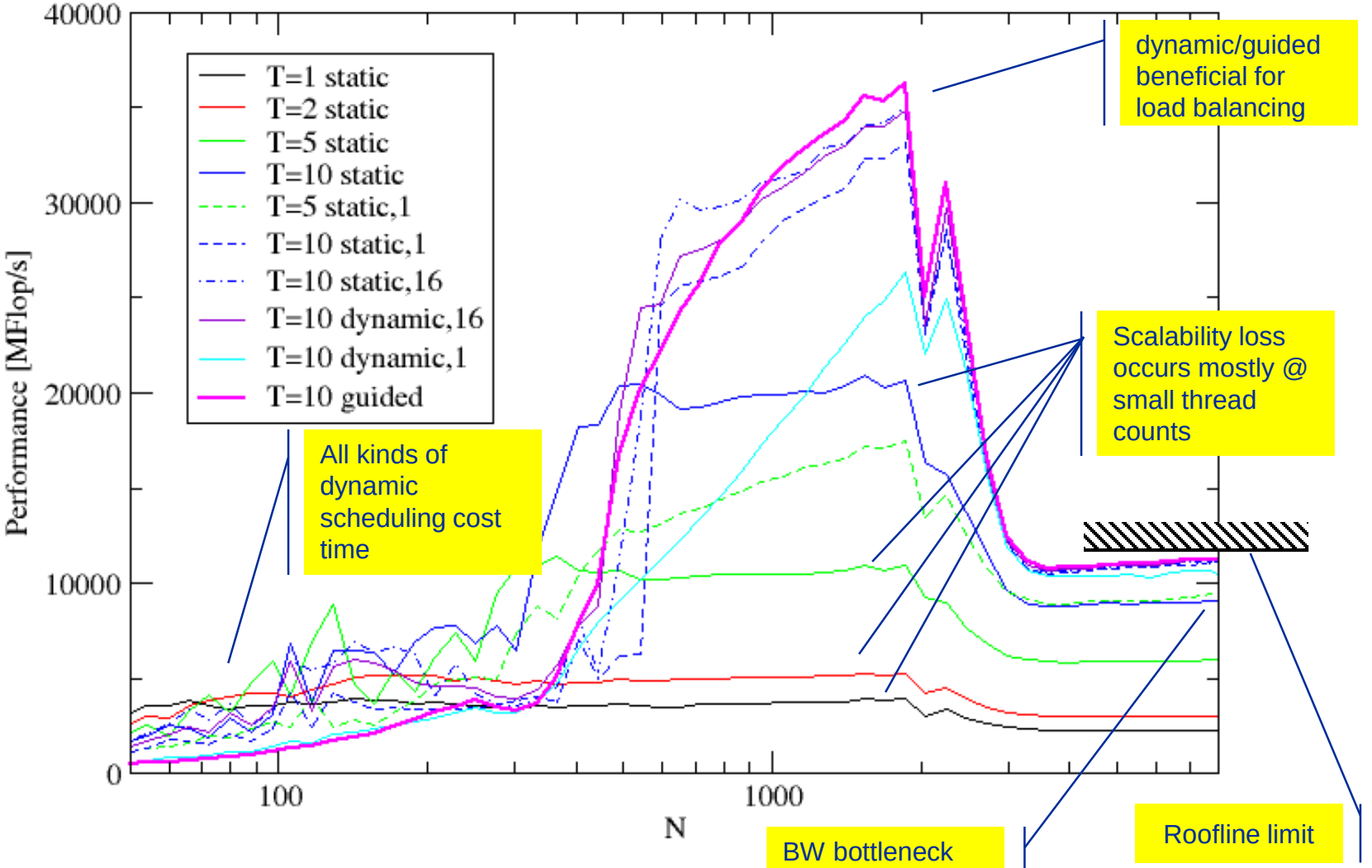
$$B_C^{mem} = 4\ \mathrm{B/F}\ \text{(DP)}$$

$$b_S = 47\,\frac{GB}{s}\ \text{(read-only)}$$

$$\rightarrow P = \frac{b_S}{B_c} = \frac{47\frac{GB}{s}}{4\frac{B}{F}} = 11.8\,\frac{GF}{s}$$

- **Several overlapping effects:**
  - Load imbalance
  - Bandwidth saturation (this is where Roofline applies!)
  - Prefetcher madness
  - OpenMP overhead

Assignment 7 – Task 1



Assignment 7 – Task 2
Speedups with accelerators

- Systems

	Peak performance (DP)	Memory bandwidth
Dual-socket Intel Ivy Bridge @ 2.2 GHz	2.2 GHz x 20 cores x 8 Flops/cy = 352 Gflop/s	2 x 40 Gbyte/s = 80 Gbyte/s
Intel Xeon Phi	1 Tflop/s	160 Gbyte/s
Nvidia K20	1.4 Tflop/s	160 Gbyte/s

Assignment 7 – Task 2
Speedups with accelerators

a) STREAM Add: `a[i] = b[i] + c[i]`

→ Speedup 2x for either accelerator (assuming NT stores)

b) Dense matrix-matrix multiply speedups:

- Phi: 1000/352 x 0.85/0.9 = 2.7
- K20: 1400/352 x 0.85/0.9 = 3.75
- Serial, non-SIMD CPU code: peak = 352/20/4 Gflop/s = 4.4 Gflop/s

→ speedup = **214** for Phi and **300** for K20

c) STREAM Add including data transfer via PCIe 2.0

→ effective BW ≈ 6.0 Gbyte/s for both accelerators

→ speedup vs. SNB = (6 Gbyte/s) / (80 Gbyte/s) = 0.075 ☺

Assignment 7 – Task 3: Raytracer

- **OpenMP-parallel raytracer: Parallelize outer loop**

```
#pragma omp parallel private(tile)
{
  tile=(char*)malloc(tilesize*tilesize*sizeof(char))

#pragma omp for private(xc,i) schedule(runtime) collapse(2)
  for(yc=0; yc<ytiles; yc++)
    for(xc=0; xc<xtiles; xc++) {
        /* calc one tile */
        calc_tile(size, xc*tilesize, yc*tilesize, tilesize, tile);
        /* copy to picture buffer */
        for(i=0; i<tilesize; i++) {
            tilebase=yc*tilesize*tilesize*xtiles+xc*tilesize;
            memcpy((void*)(picture+tilebase+i*tilesize*xtiles),
                   (void*)(tile+i*tilesize),
                   tilesize*sizeof(char));
          }
      }
}
```

Raytracer sequential optimizations

- Avoiding FP divides in `shade()` and `calc_tile()` yields about 15% speedup

```
r = 1./sqrt(nx * nx + ny * ny + nz * nz);
nx *= r; ny *= r; nz *= r;
...
r = sqrt(ldx * ldx + ldy * ldy + ldz * ldz);
rr = 1./r;
ldx *= rr; ldy *= rr; ldz *= rr;
...
r = 1./sqrt(dx * dx + dy * dy + dz * dz);
c = 100 * shade(2.1, 1.3, 1.7, dx * r, dy * r, dz * r, 0);
```

OpenMP-parallel raytracer performance on Ivy Bridge 2.2 GHz (Emmy)