Lab Course
Efficient Programming of Multicore Processors and Supercomputers

Report 3: Auto Parallelization & OpenMP

Jonas Mayer, Paul Preißner, Konrad Pröll

Fakultät für Informatik
Technische Universität München

June 1th 2017

Task 3.2 Automatic Parallelization - Getting it to work

- flag `-parallel`: enables auto parallelization
- flag `-opt-report-phase=par`: compilation yields report about parallelized parts, parts that could not be parallelized and the underlying reasons
- BUT: reports no loops were changed due to assumed data dependencies
- `#pragma ivdep`: ignores dependencies for a single loop
- BUT: still no parallelization. Turns out icc has a threshold for optimizations it thinks are not worth it.
- flag `-par-threshold0`: modifies (0=removes) this threshold
- alternatively `#pragma parallel always`: overrides threshold for a single loop
- flag `-guide-par`: returns possible manual optimizations during compilation did not find anything for relax_jacobi.c

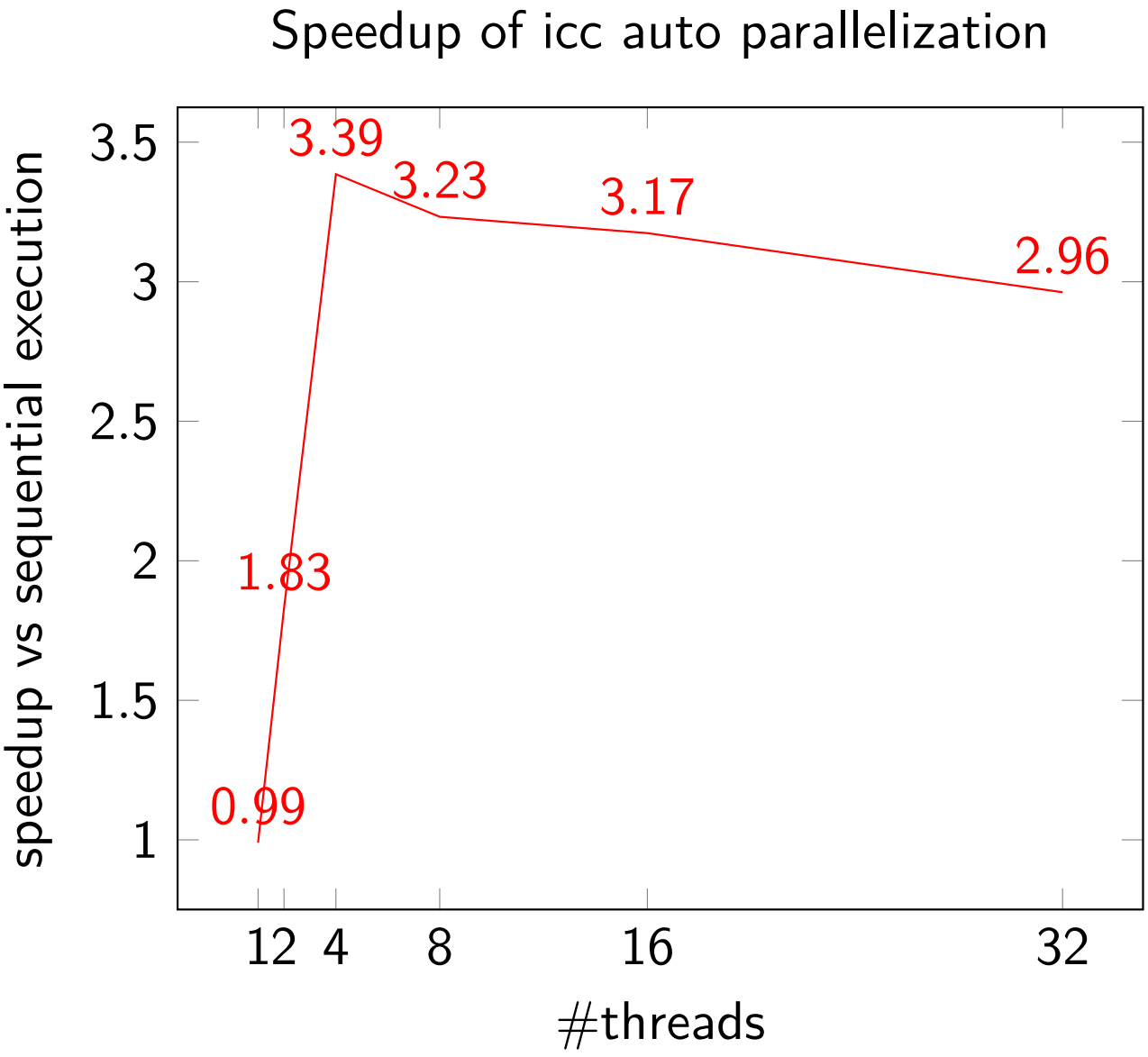
Task 3.2 Automatic Parallelization - pragma snippet

```
1  double relax_jacobi( double **u1, double **utmp1,
2            unsigned sizex, unsigned sizey )
3  { [...]
4  #pragma parallel always
5    for( i=1; i<sizey-1; i++ ) {
6           int ii=i*sizex;
7           int iim1=(i-1)*sizex;
8           int iip1=(i+1)*sizex;
9
10 #pragma ivdep
11       for( j=1; j<sizex-1; j++ ){
12          unew = 0.25 * (u[ ii+(j-1) ]+
13                                      u[ ii+(j+1) ]+
14                                      u[ iim1+j ]+
15                                      u[ iip1+j ]);
16                      diff = unew - u[ii + j];
17                      utmp[ii+j] = unew;
18                      sum += diff * diff;
19
20       }
21   } [...]
22 }
```

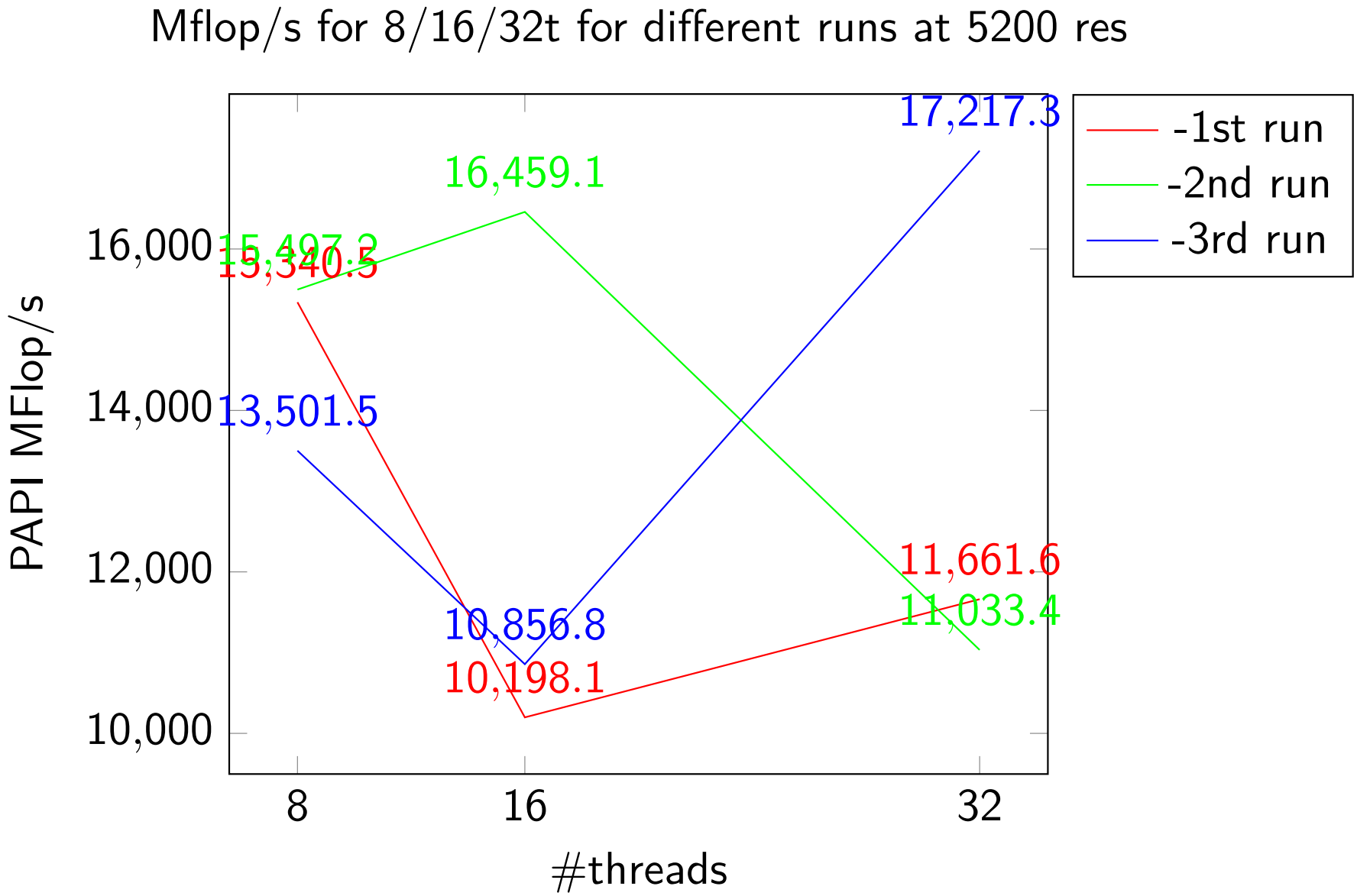
Task 3.2 Automatic Parallelization - Speedup 1/2

- Do performance measurements for 1, 2, 4, 8, 16 and 32 threads on SuperMUC [...] for problem size 5200. [...] sequential code as the basis for the speedup calculation.
  See next slide.
- Is there a difference between the sequential time from Assignment 3.1 and the sequential time of the OpenMP version?
  No. At least no significant difference.

Task 3.2 Automatic Parallelization - Speedup 2/2



Task 3.2 - Curious performance with many threads



Task 3.2 - Curious performance with many threads

- Performance with 8 to 32 threads tends to fluctuate.
- Possible explanation:
  - We run on Intel Xeon Processor E5-2680 CPUs, each has 8c/16t, thus any run with >8 threads uses Hyper-Threading
  - "Computational intensive applications with fine-tuned floating-point operations have less chance to be improved in performance from Hyper-Threading, because the CPU resources could already be highly utilized."
  - "Cache-friendly applications might suffer from Hyper-Threading enabled, because logical processors share the caches and thus the processes running on the logical processors might be competing for the caches' access, which might result in performance degradation."
  - Source: `https://www.researchgate.net/publication/267242498_An_Empirical_Study_of_Hyper-Threading_in_High_Performance_Computing_Clusters`

Task 3.3

- 1.

Thank you for your attention!