Performance Considerations

ICHEC
Irish Centre for High-End Computing

ICHEC
Irish Centre for High-End Computing

Parallelisation Approach

- Profile your code
- Determine the most time-consuming parts.
- Decide if those parts can be parallelisible.
- Add OpenMP directives and clauses in the code.
- Check correctness and measure the time.

- It may be easy to write a correctly functioning OpenMP program.
- But, it is not so easy to create a program that provides the desired level of performance!

ICHEC
Irish Centre for High-End Computing

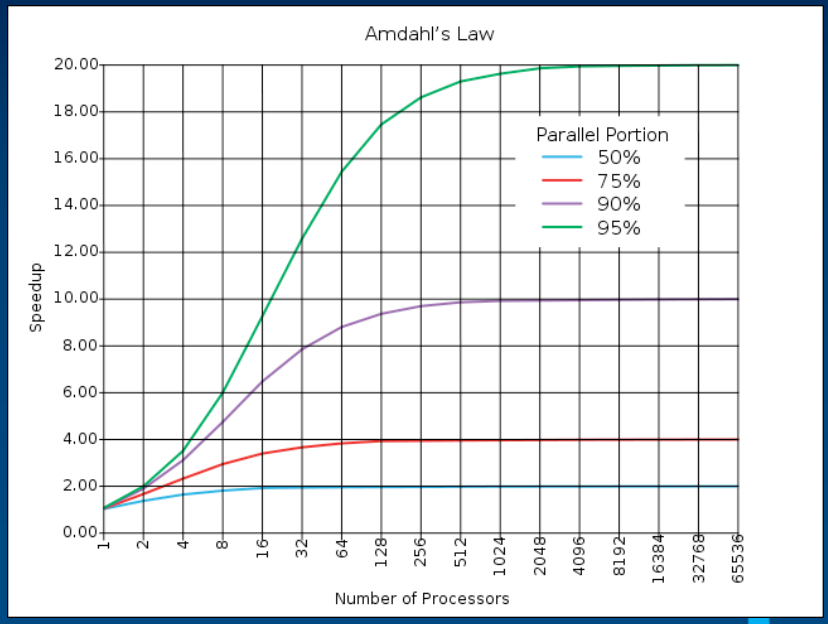
Overhead …(1)

If a parallelized loop does not perform well, consider

- Sequential overhead. Amdahl' Law:

The speedup is limited by the sequential fraction of the program.

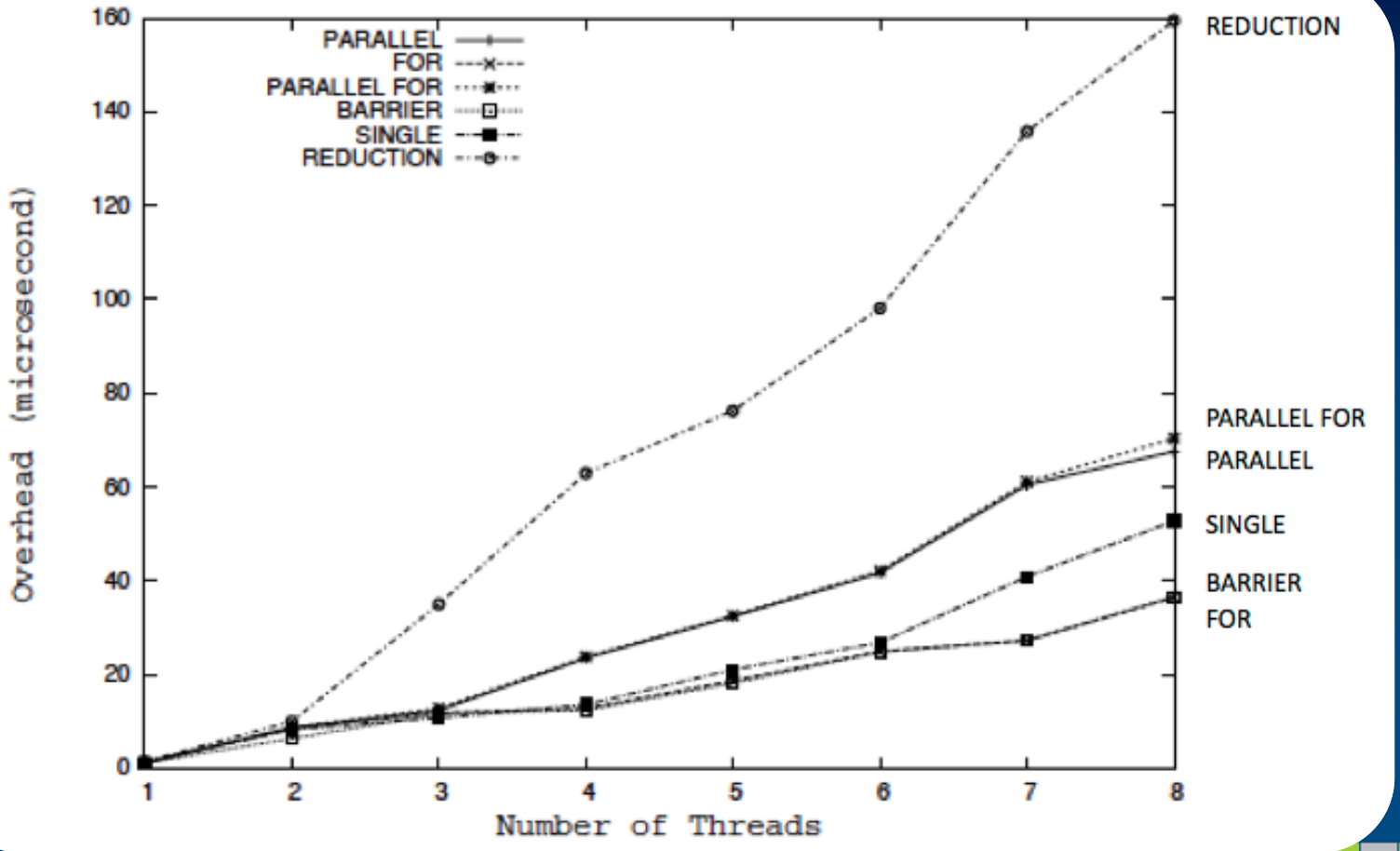
$$\frac{1}{f + \frac{1-f}{p}}$$



- Software overhead imposed by parallel compilers, operating system etc.
- Parallelisation Overhead: Thread start up costs, The amount of time spent handling OpenMP constructs, Thread termination time.

ICHEC
Irish Centre for High-End Computing

OpenMP Directives Overhead



Barbara Chapman, Gabriele Jost and Ruud Van Der Pas, Using OpenMP: Portable Shared Memory Parallel Programming, Volu
10, MIT Press, 2008.

ICHEC
Irish Centre for High-End Computing

Overhead …(2)

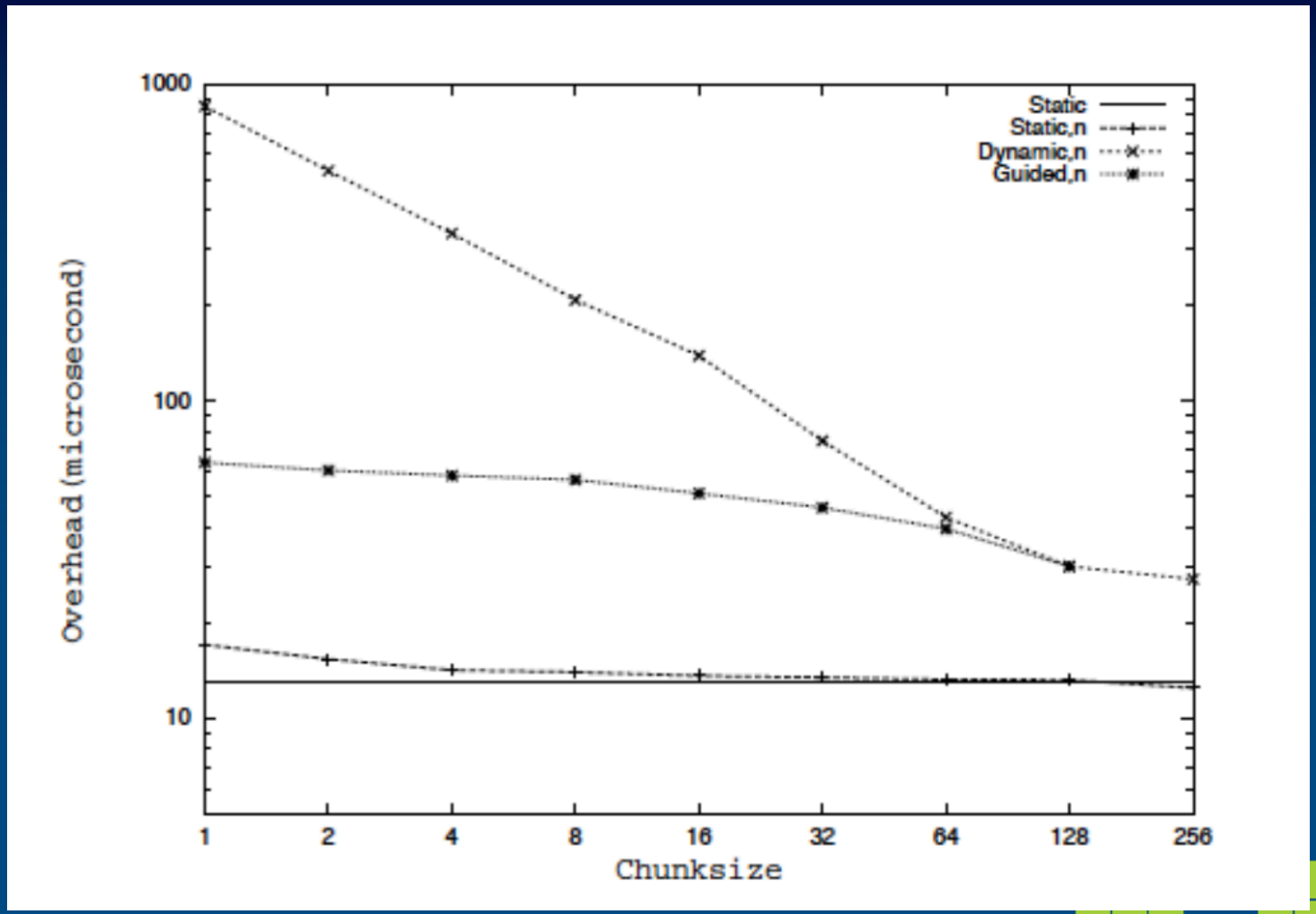
- Parallel overhead at low iteration counts. Avoid by use of if clause.

```
#pragma omp parallel for if(M > 800)
for(j=0; j< M; j++)
    aa[j] = alpha*bb[j] + cc[j];
```

- Many references to shared variables. Use private data, allocated on stack.
- Unnecessary synchronization; Large Critical Regions; prefer atomic update if possible. Use nowait clause.
- Load imbalances: Unequal work loads lead to idle threads and wasted time. Use proper scheduling type.

ICHEC
Irish Centre for High-End Computing

OpenMP Scheduling Overhead



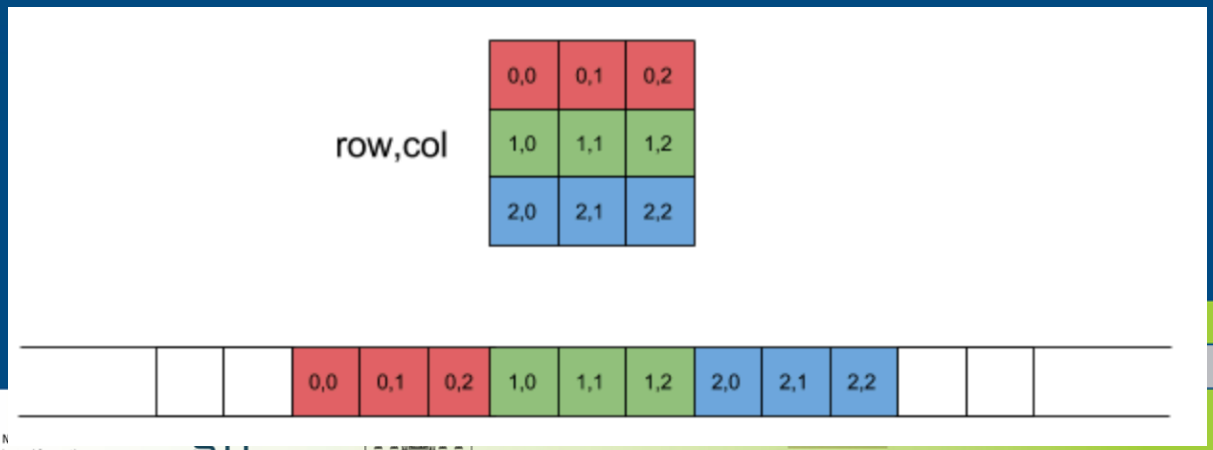
Barbara Chapman, Gabriele Jost and Ruud Van Der Pas, Using OpenMP: Portable Shared Memory Parallel Programming, Volume 10, MIT Press, 2008.

ICHEC
Irish Centre for High-End Computing

Overhead …(3)

- Low cache reuse.
  - organize data accesses so that values are used as often as possible while they are still in the cache.
    - In C, a 2-D array is stored in rows (and in columns in Fortran). Array is accessed along the rows:

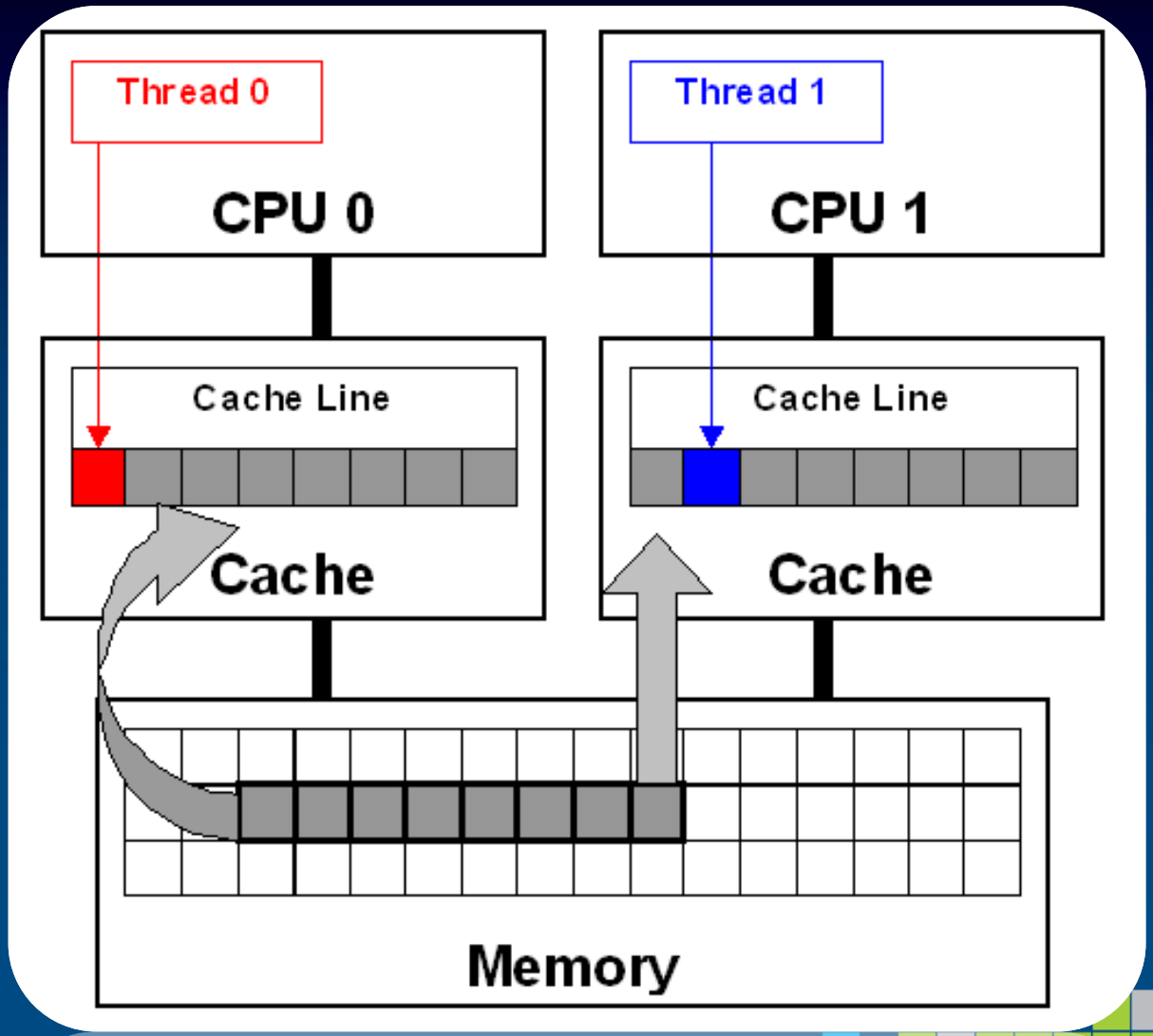
```
for(i=0; i<n; i++)
  for(j=0; j<n; j++)
    sum+=a[i][j];
```



ICHEC
Irish Centre for High-End Computing

- false sharing: two or more threads repeatedly update the same cache line.

This invalidates the cache line and forces a memory update to maintain cache coherency.



ICHEC
Irish Centre for High-End Computing

More Recommendations

- Maximize parallel regions, Minimize serial code
- Remove dependencies among iterations
- Parallelize outer loops
- Minimize the number of directives
- Make sure speedup in parallel region enough to overcome overhead
  - Is number if iterations in loop large enough?
  - Is amount of work per iteration enough?

ICHEC
Irish Centre for High-End Computing

Debugging your Code …(1)

- Typical signs that your program needed debugging:
  - It fails to complete (crashes)
  - It produces incorrect output (!##%?)
  - It fails to progress (hangs)

- Diagnosing the problem:
  - pay attention to compiler warnings
  - inspect the job exit code
  - look at the job output file
  - using print statements

ICHEC
Irish Centre for High-End Computing

Debugging OpenMP Code …(2)

- Shared memory parallel programming opens up a range of new programming errors arising from unanticipated conflicts between shared resources.

- Race conditions:
  - Multiple threads are updating the same shared variable simultaneously.
  - Hard to find, not reproducible, answer varies with number of threads.

- Deadlock:
  - When threads hang while waiting on a locked resource that will never become available.
  - A simple approach is to put print statement in front of all lock calls.

ICHEC
Irish Centre for High-End Computing

Profiling and Debugging Tools

- Profiling Tools:
  - GNU profiler: gprof
  - Allinea MAP
  - Intel VTune
  - ompP
  - Tau
  - VampirTrace
- Debugging Tools:
  - GNU debugger: gdb
  - Intel Inspector
  - Valgrind
  - DDT
  - Totalview