|  |  |  |
| --- | --- | --- |
| **ACTIES OP HET SCHERM** | **VOICE-OVER** | **DUUR** |
|  | Modern CPUs have many cores, each capable of executing an independent instruction thread. However, your program needs to be designed to take advantage of that potential. One of the easiest ways to accomplish this for scientific computing is OpenMP. You annotate your source code with directives, and the compiler will generate the low-level code to spawn threads and divide the computations over them. This is a very extensive topic, and PRACE has some excellent courses dealing with OpenMP, so this is just to give you the flavor. |  |
| 1. Show compute\_pi.f90 | 1. Explain algorithm 2. Point out opportunity for parallelization |  |
| 1. Compile and run timed |  |  |
| 1. Show compute\_pi\_omp.f90 | 1. Open parallel region 2. Explain shared versus private 3. Explain functions, point out use statement 4. Explain omp do 5. Explain reduction |  |
| 1. Compile and run timed | 1. Discuss performance gains |  |
| 1. Show speedup, parallel efficiency | 1. Discuss parallel efficiency |  |
|  | Take into account that this was a really simple example. In practice OpenMP is not that easy to use. You have to be careful not to introduce race condition, and getting good parallel efficiency requires work. |  |
| **TOTALE DUUR** | | *Maak je screencast niet langer dan ca. 6 minuten.* |