Experience with the parallel execution of SERPENT2

J. Eduard Hoogenboom

Delft Nuclear Consultancy

Parallelisation

- In general two options:
- MPI (Message Passing Interface)
- OpenMP
  - ✓ both applicable to C/C++ and Fortran

- PVM (Parallel Virtual Machine) is outdated

MPI

- provides a set of routines for communication between processors/cores
- identical programs start on all selected computer cores
  - mpiexec –n 8 executable [arguments]
- memory limitations for large problems
- MPI package available from ANL as MPICH2
- alternative: Open MPI
  - provides a more modern implementation

OpenMP (threading)

- ✓ easier to implement
- ✓ executes part of program on multiple threads
- ✓ uses shared memory
- ✓ can be used on one computer node only
- ✓ activated by compiler option -openmp
- ✓ uses compiler directives like: #pragma omp …
- ✓ can be used in combination with MPI

SERPENT

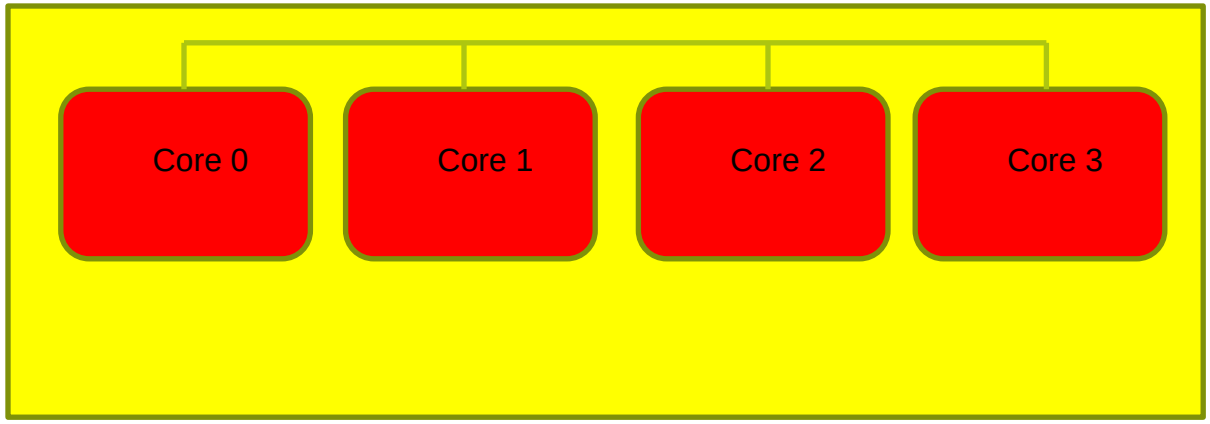
- SERPENT1 used MPI only
- SERPENT2 allows both MPI and openmp
- mpiexec –n 4 sss2 –omp 8 input_file
  - uses in this case: 4 x 8 processor cores
- however,
  efficiency depends on your computer system

Compiling SERPENT

- SERPENT will be compiled and built with make command
- make has an option for parallel compilation:
- make –j <n>
  - with n number of processor/cores to be used
- Advice from MCNP:
  - ✓ You may take n twice the number of available cores

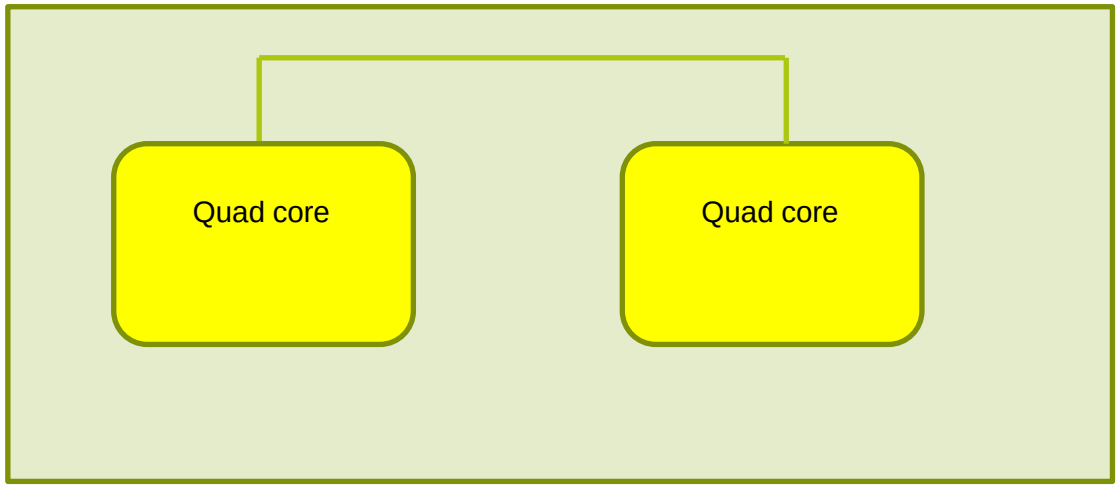
Computer nodes, processors and cores

Quad core processor



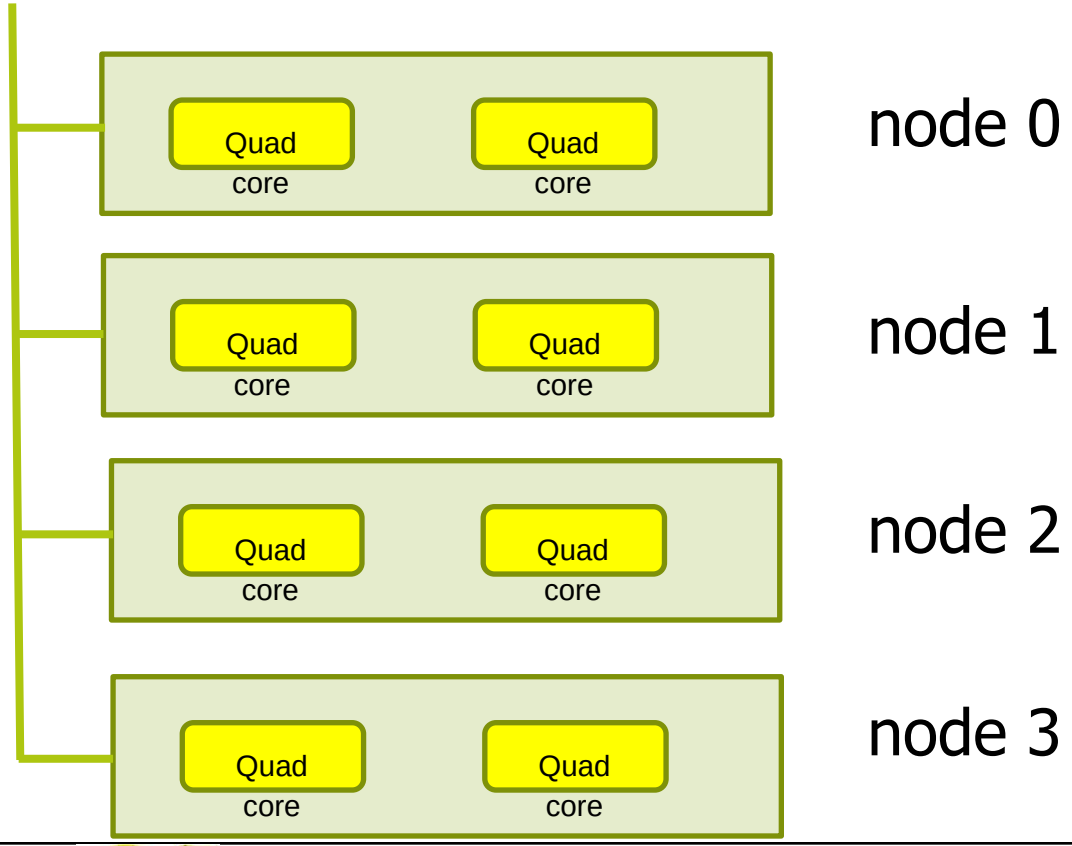
Computer nodes, processors and cores

Computer with 8 cores



Computer nodes, processors and cores

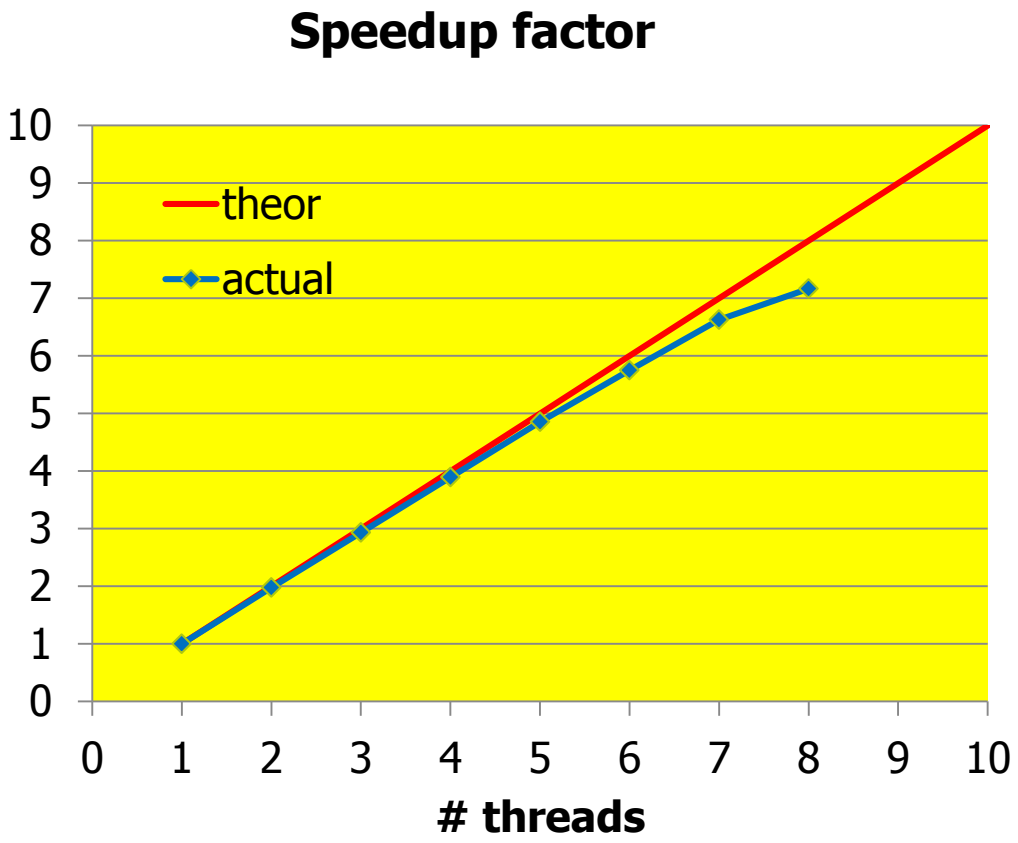
Computer cluster with 4 nodes x 8 cores



Test problem

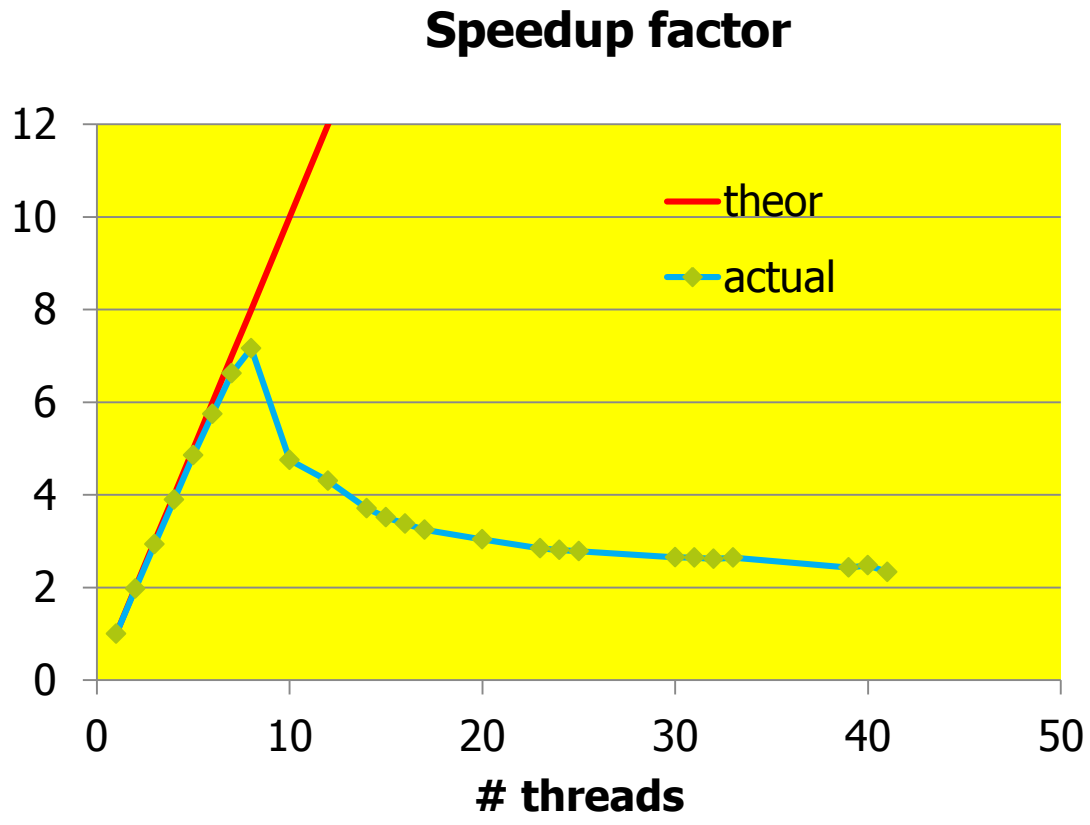
- SERPENT2 calculations for a BWR pin cell
- with 10,000 histories/batch and 500 batches
- varying number of MPI cores and threads
- determine speedup $$SU = \frac{T_{1\,core}}{T_{n\,cores}}$$
- using TRANSPORT_CYCLE_TIME only

Results with OpenMP

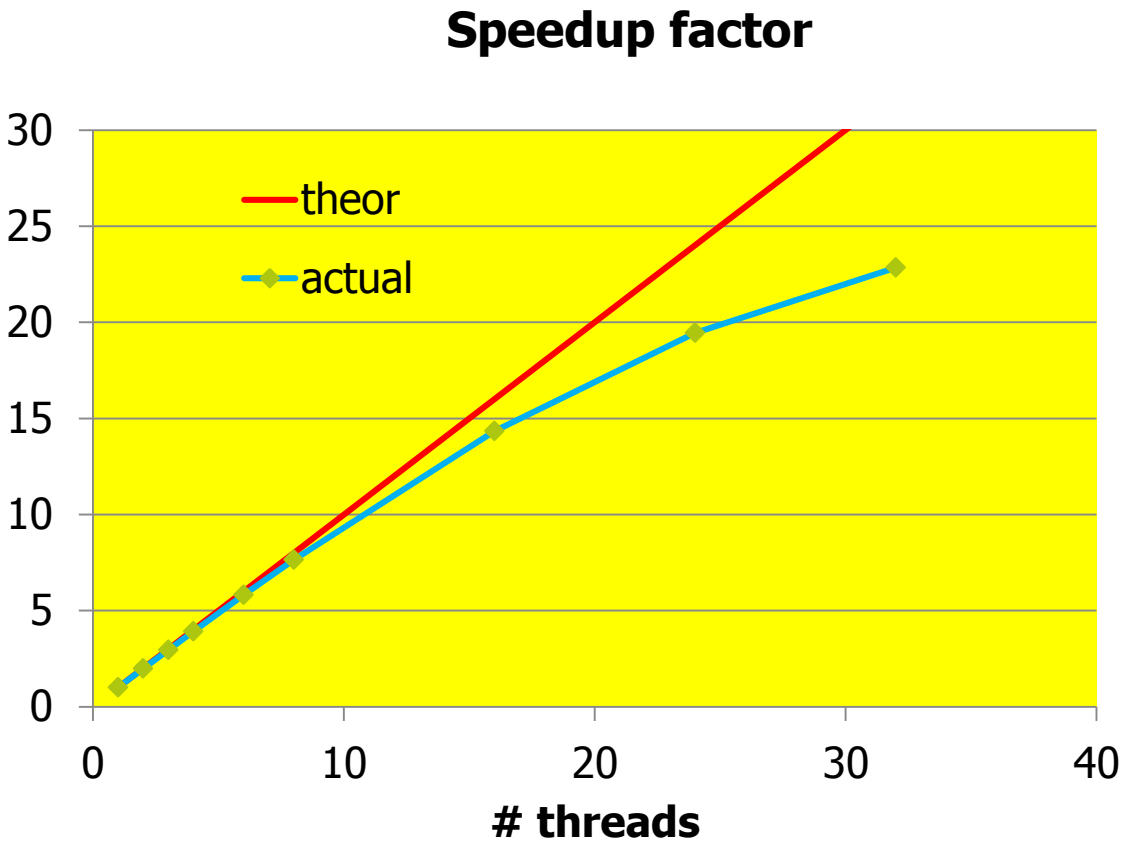


Speedup on a node with 8 cores

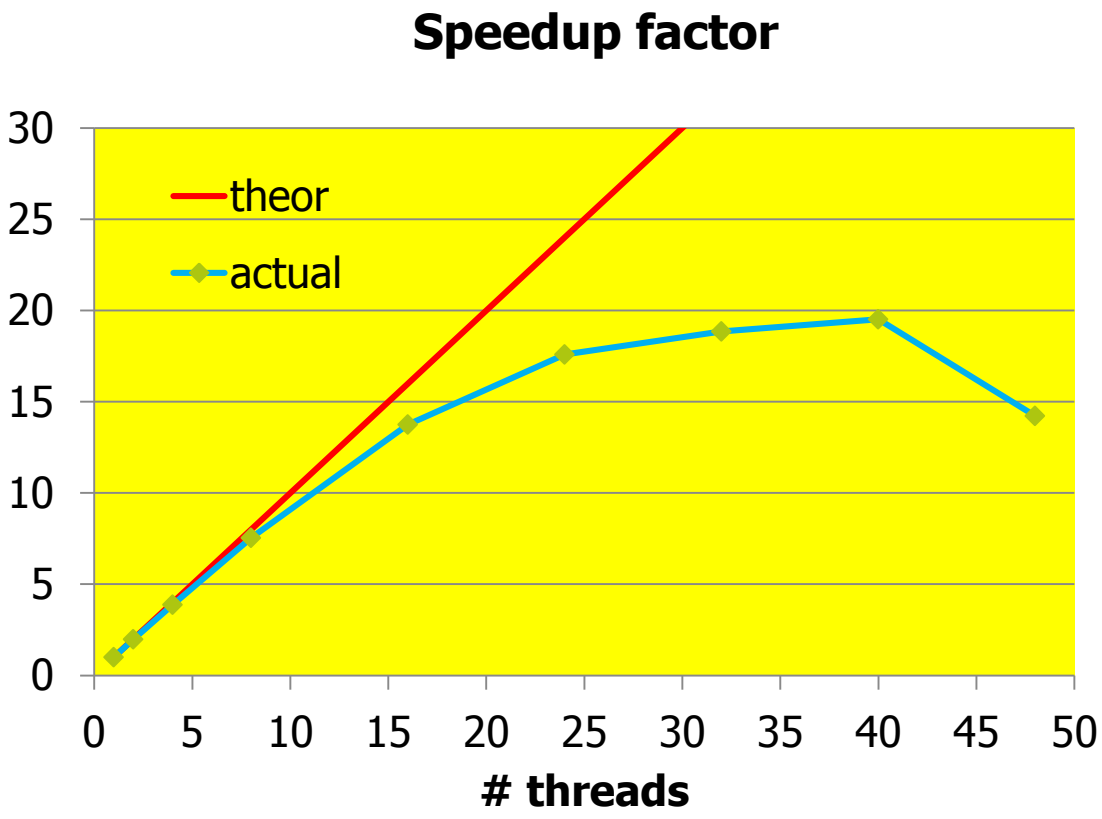
Using more threads than cores



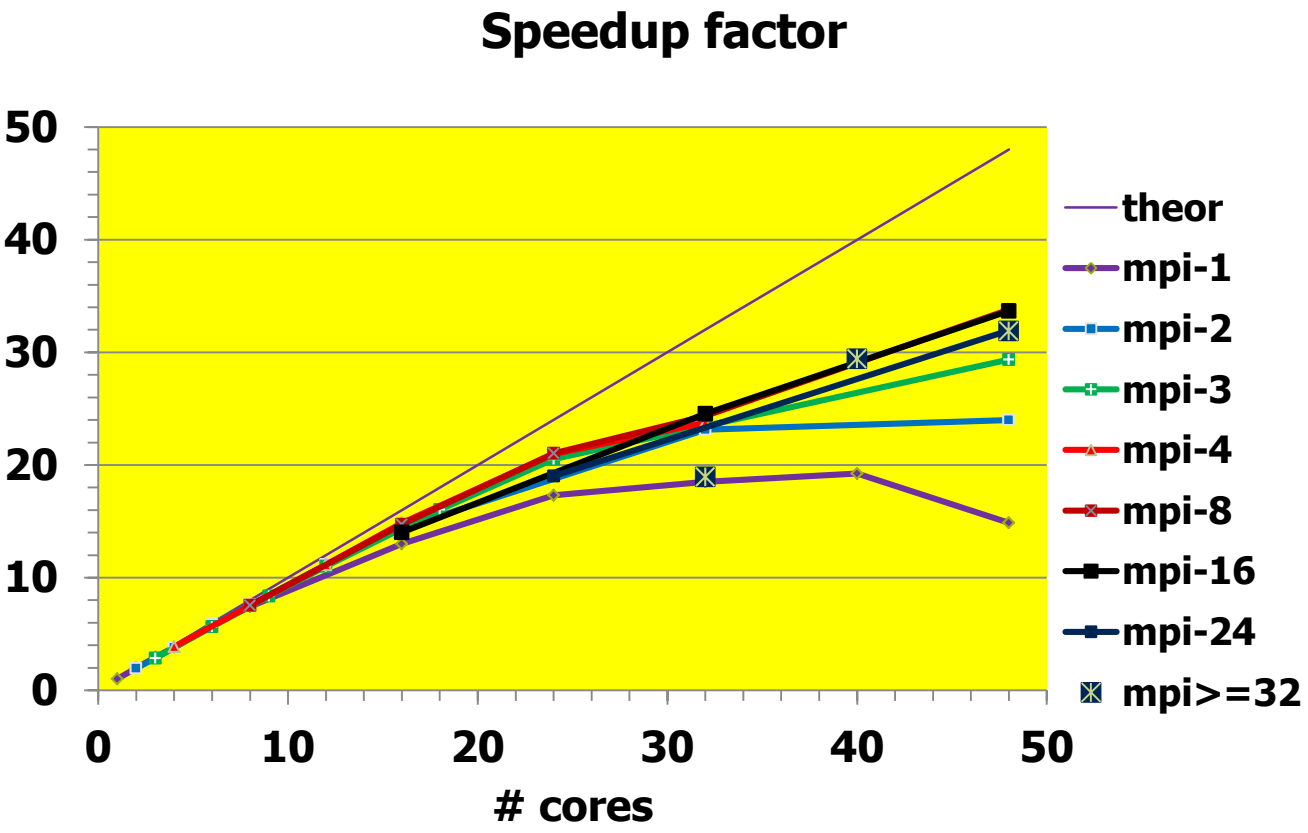
Speedup on a node with 32 cores



Speedup on a node with 48 cores



Combinations of MPI and threads



Combinations of MPI and threads

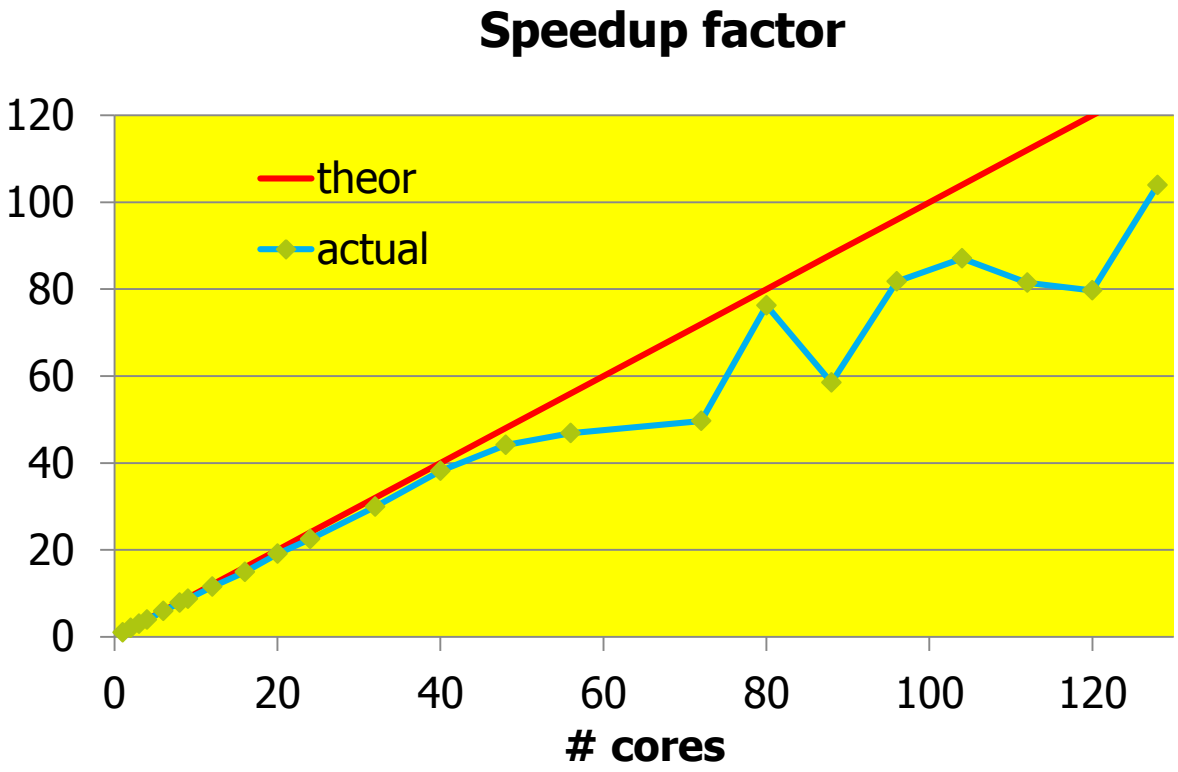
- If others have access to the node:
  - Claim total number of cores needed via scheduling parameters
  - Specify number of MPI cores with mpiexec
  - Specify number of threads with -omp

Using more than one node

- If you have, e.g., 4 nodes available, each with 8 cores
- specify at maximum 8 threads
- So, you may specify 4 MPI tasks with 8 threads each
- However, scheduler may assign 4 MPI tasks to cores on the *same* node!

➢ Won't give expected speedup

More than one node with MPI only



- Nodes with 32 cores

Using more than one node

- On (specific) supercomputer:
- you always get all cores on a node
- Hence, tell PBS scheduler to use 4 nodes with 1 core each:
  - ➢ #PBS -l nodes=4:ppn=1
  - ➢ in job file: mpiexec –n 4 sss2 –omp 8 …
  - ✓ uses 8 threads on each node
- Whether this works efficient, depends on internode communication ➢ Infiniband

Conclusions

- SERPENT2 provides flexible options for parallel execution
- On a single computer node:
  - ✓ openMP can well be used
  - ✓ don't use more threads than cores available
  - ✓ use combination of MPI and openMP for larger number of cores
- Take care of scheduling parameters
- Parallel execution with very large numbers of cores will not be efficient

Acknowledgement

Research supported by the European Union project HPMC

High-Performance Monte Carlo