LA-UR-21-26619
Approved for public release; distribution is unlimited.

Title: Parallelism in MCNP® 6.2

Author(s): Bull, Jeffrey S.

Intended for: 2021 MCNP® User Symposium, 2021-07-12 (Los Alamos, New Mexico, United States)

Issued: 2021-07-13 (Draft)

Los Alamos NATIONAL LABORATORY

Parallelism in MCNP® 6.2

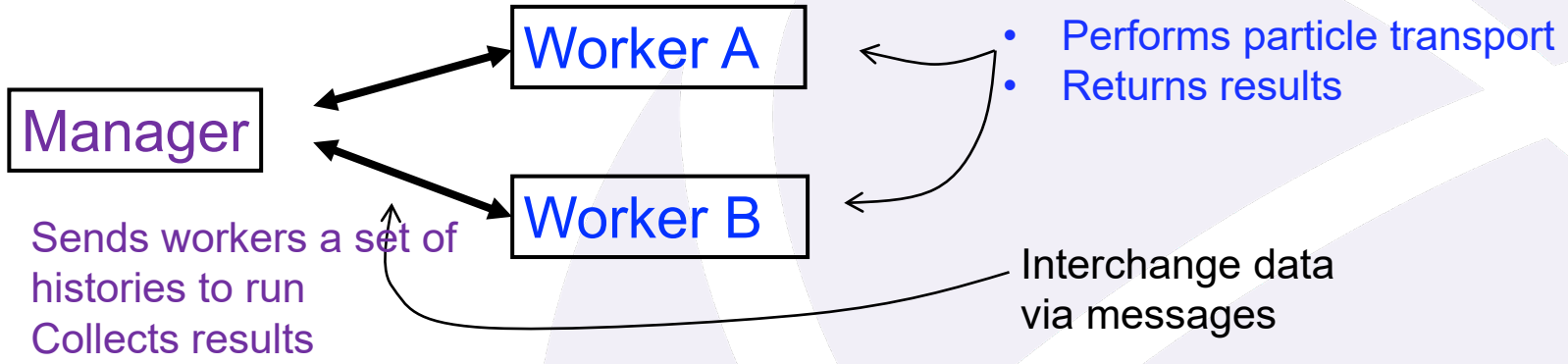
Jeffery Bull
XCP-3 (Monte Carlo Codes

Roundtable on MCNP Parallelism Performance
2021 MCNP® User Symposium

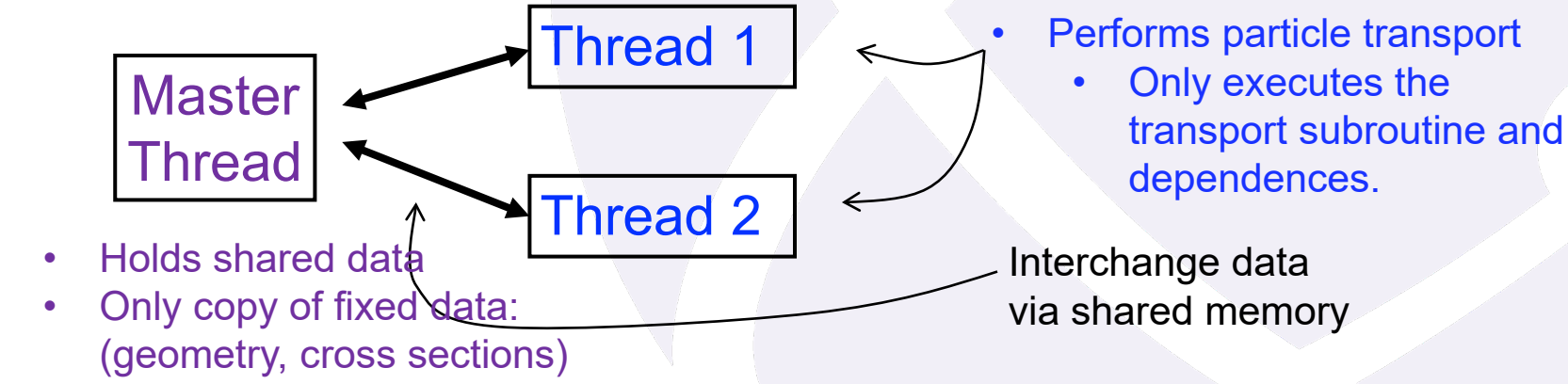
July 16, 2021

MCNP uses two methods to run in parallel

Message Passing Interface (MPI)



Shared memory multiprocessing (OpenMP)



Trade Offs

- **MPI**
  - Pros
    - Easier to implement
    - Can be use with (almost) all features of MCNP6
    - Only way to run on multi-node clusters.
  - Cons
    - Implementation on Linux and MacOS systems require user to compile MCNP

- **OpenMP**
  - Pros
    - Included in the distributed executables.
    - Limited to a subset of MCNP6 capabilities
  - Cons
    - Difficult to implement
      - Some sections in the parallel region must be run serially – requires thread locks
      - Insure individual threads don't overwrite critical data
    - Limited to a subset of MCNP6 capabilities
    - Speedup depends on computer architecture (NUMA memory)

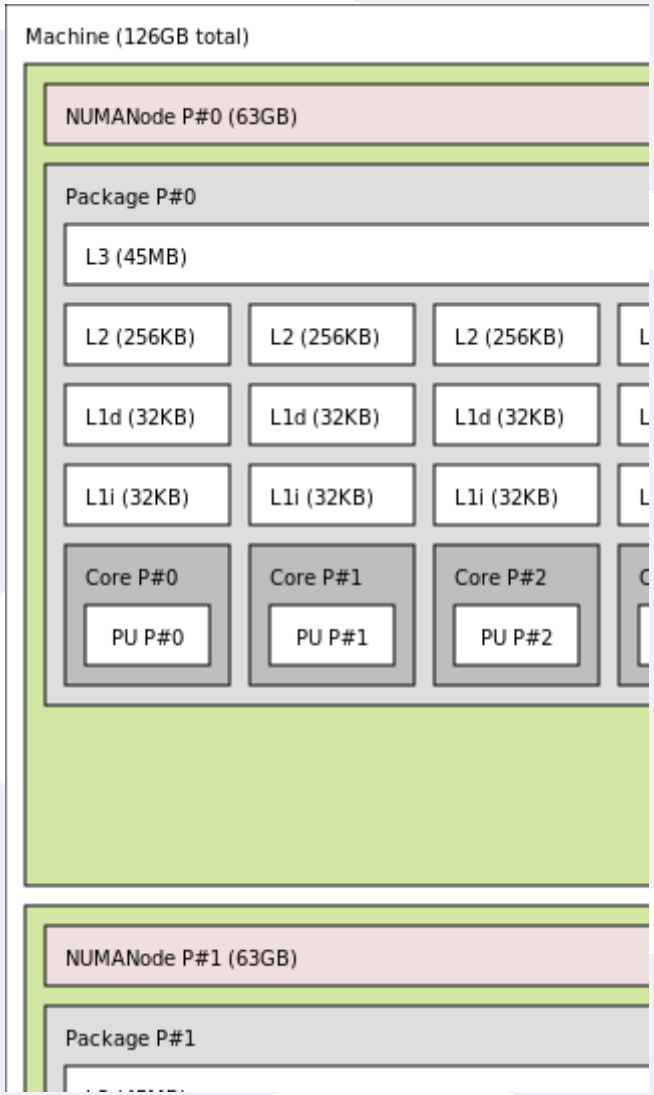
Hand off to Avery

OpenMP Performance Of The Test Problem

Snow cluster

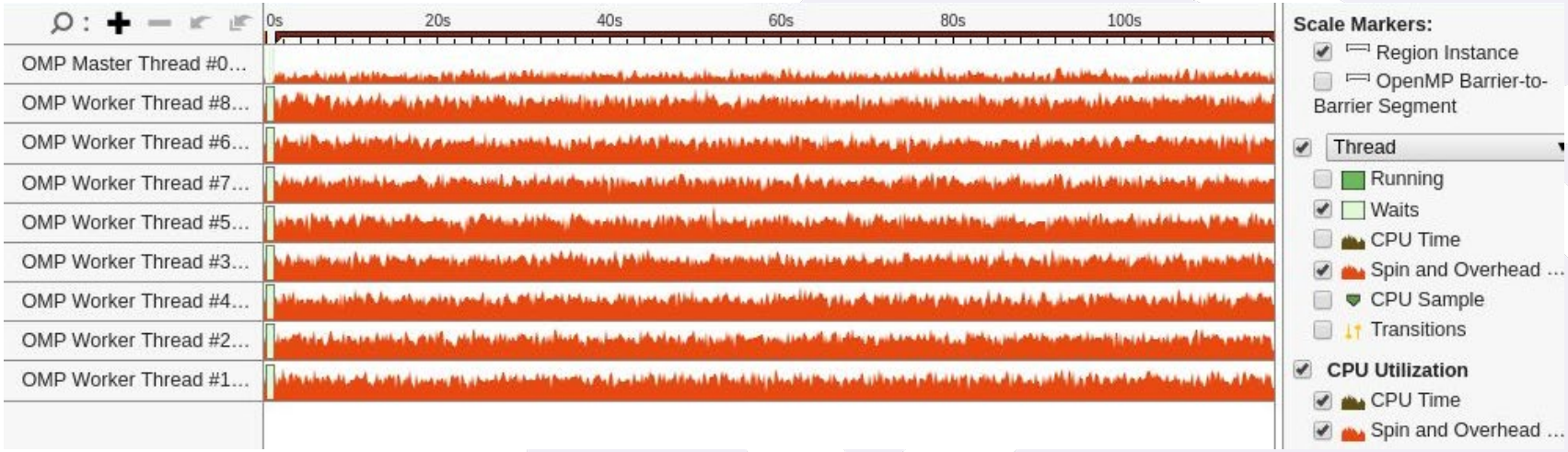
- 128 GB per node
- 2 sockets/node
- 18 CPUs/socket
- No hyperthreading
- Non-uniform memory access (NUMA)

Compare results for 9 and 36 threads.



Fraction Of Time Thread Is Waiting For Work

9 threads: 55% CPU time spent in spin/overhead state



36 threads: 90% CPU time spent in spin/overhead state