Multithreaded Parallelization

Guillaume Seguin
guillaume.seguin@ens.fr

INRIA

Workshop on ns-3, March 15th, 2010

Requirements

- Multicore shared memory systems: at least 8 cores, target 24, potentially more.
- Transparent for the user: no partition specification
- Should go faster
- Work with wireless simulations

MPI: the traditional approach

- One process per partition
- Static partition <-> node mapping: no data/code migration
- User must create partition map by hand

Multithreaded parallel simulation

- Degenerate partition map: one node <-> one partition
- Classic conservative synchronization algorithm with lookahead
- One worker thread per physical cpu:
    - take partition from worklist
    - execute partition until conservative boundary

Pros:

- No partition specification
- automatic load balancing across CPUS
- measurable speedup

Challenges

- Overhead of thread-safe APIs:
    - Events crossing partitions
    - Copy packets crossing partitions
    - Reference counting (atomic ops too slow)
- Barrier implementation:
    - Decrease latency: spin locks
    - Tree barrier vs global counter vs posix barrier vs posix cond var

Results

- Thread-safe refcounting is costly: need to write paper about it
- Must implement wifi RxTxTurnaround support for efficient wifi.
- It goes faster !
- It's transparent !
- Multicore shared memory systems are very hard to program efficiently

References

- *Good news for parallel wireless network simulations*, P. Peschlow: same approach, use of rxTxTurnaround time modeling to speedup wireless simulations
- *Multi-core parallelism for ns-3 simulator*: internship by Guillaume Seguin (http://guillaume.segu.in/papers/ns3-multithreading.pdf)