Multi-core parallelism for ns-3

Guillaume Seguin

INRIA Sophia-Antipolis Méditerranée, PLANETE team

16/09/2009
1. Doing parallelism in ns-3
   - Goals
   - About ns-3

2. Being non-intrusive
   - Partitioning the network
   - Synchronization algorithm
   - Thread safety

3. Speeding up simulations
   - Optimizing barriers
   - Thread-safe reference counting
   - Solving memory contention and balancing workload

4. Further work
Goals

- Being non-intrusive
- Speeding up simulations
ns-3 is a discrete-time, event-driven network simulator

ns-3 is split into a simulator part, which has no clue about the fact that it simulates networks, and a models part, which holds all the actual networking knowledge

Network topologies and applications are described by C++ or Python scripts
We do not want to ask the user to partition the network

We cannot compute an efficient partitioning (we need more than the network topology)

→ We will use partitions with only one node
What does ensure simulation consistency? An ordering constraint on events dates at each node.

- Conservative algorithms: enforce ordering constraint, simple but maybe slower
- Optimistic algorithms: does not enforce the constraint at processing time, requires extra coding (user-level, compiler-level, simulation status snapshots)

Our choice: conservative algorithms.
Synchronization barrier based algorithm

**Lookahead**

The lookahead for a given partition is the minimum delay before a neighboring partition may receive an event from this partition.

At each iteration:

- Threads gather at a meeting point (the barrier)
- A global maximum date up to which events can be processed is computed (using the date of the next event of each partition + the partition lookahead)
- Threads process events of all the partitions up to this date
In ns-3, most structures are only written by the node they are related to, and are read in a thread-safe way.

Problems:

- **Reference counting**: disabling it in spots where problematic Ref/Unref could occur would be too intrusive → make it thread-safe
- **Packet-related caches and buffers**: simply disable them
Doing parallelism in ns-3
Being non-intrusive
Speeding up simulations
Further work

Optimizing barriers
Thread-safe reference counting
Solving memory contention and balancing workload

Guillaume Seguin
Multi-core parallelism for ns-3

- POSIX barriers
- Semaphores barriers
- Spinlock barriers
- Tree-shaped spinlock barriers

![Diagram showing performance comparison between POSIX, Sem, Spin, and TreeSpin barriers across different thread counts.](image-url)
Thread-safe reference counting

- Atomic operations implementation
- Thread-local reference counting
- Thread-local buffered reference counting: instead of directly applying operations, store them in a thread-local buffer and push the buffer when its full to a central processing location
Thread-safe reference counting performance

Guillaume Seguin
Multi-core parallelism for ns-3
Avoid contention on a single shared resource by dedicating each partition to a core
Balancing workload

- Dedicate a given ratio of the partitions to cores and keep the remaining in a shared set
- Use several shared sets to reduce contention even more
Results

Guillaume Seguin
Multi-core parallelism for ns-3
Further work

- Find better test cases
- Even smarter workload balancing
- Make packet-related caches thread-safe
- Computing lookahead for wireless networks
Conclusion

- Non-intrusivity leads to tough design choices
- Thread safety is quite expensive
- Linear gain looks really hard to reach
- Yet, we were able to achieve 20% speedup with 8 times more computation power.
Questions ?