AutoNUMA23
vs
sched-numa rewrite

Red Hat, Inc.

Andrea Arcangeli
aarcange at redhat.com

17 Aug 2012

redhat.

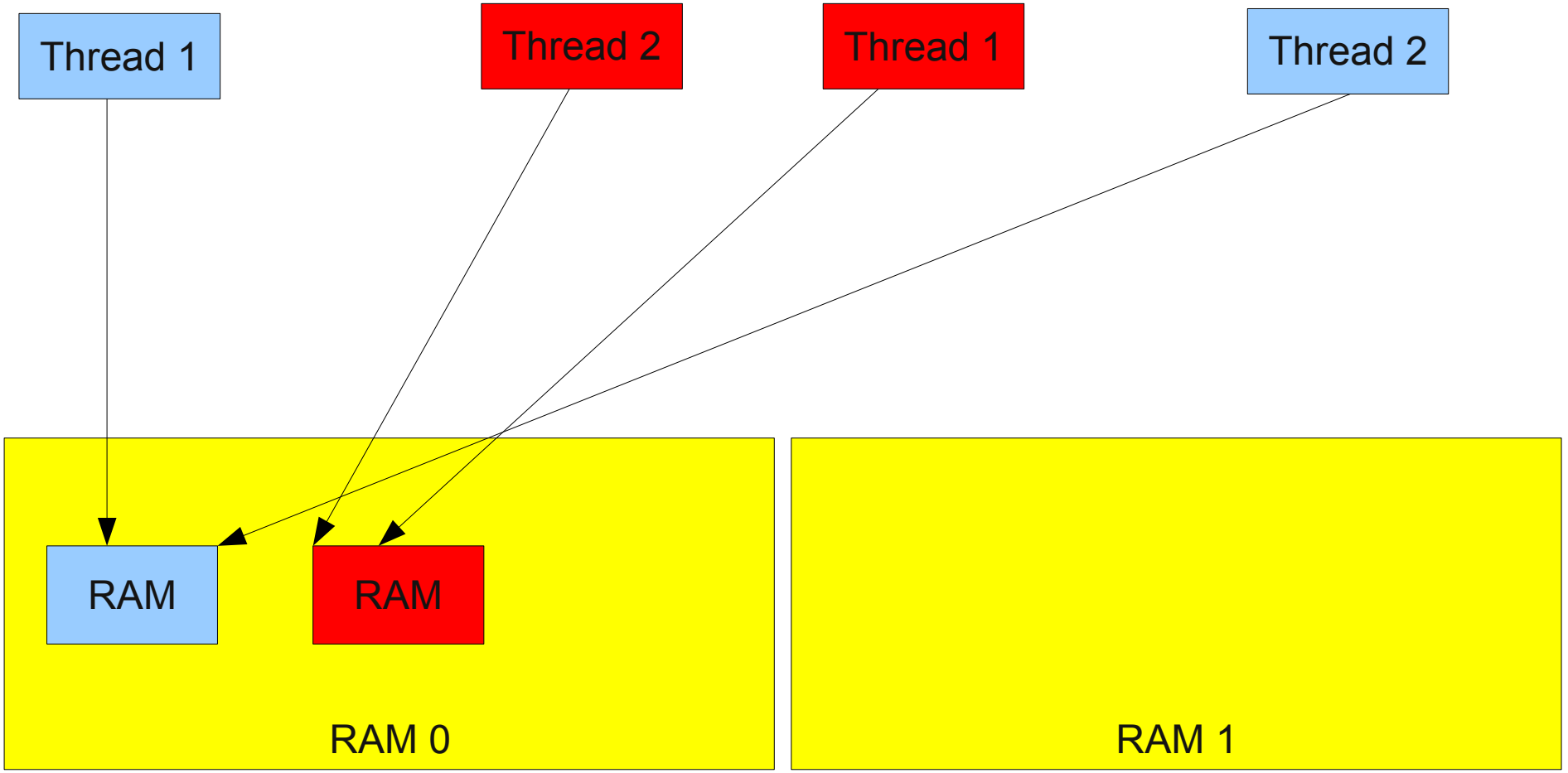
Objective

- Run the autonuma-benchmark (tag 0.1 d20e5a0) to compare:
  - AutoNUMA23 fe7de11
    - Based on 3.6-rc1 0d7614f
  - sched-numa rewrite
    - Message-Id 20120731191204.540691987 applied on 3.6-rc1 0d7614f
  - Upstream 3.6-rc1 0d7614f
- Run on 2 nodes with HT enabled
- Run on 8 nodes with HT enabled

AutoNUMA-benchmark tests

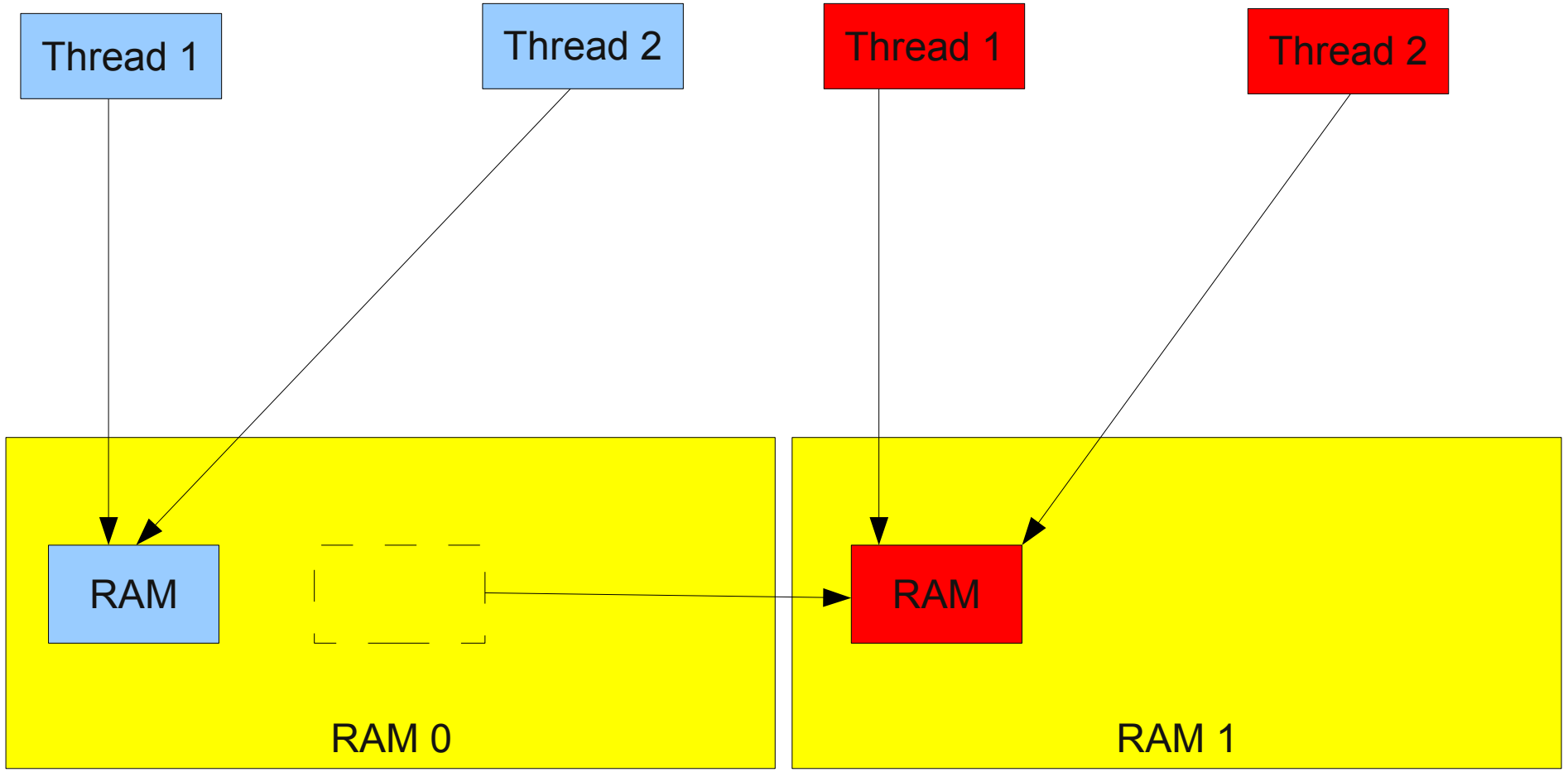
- P=# of processes, T=# of threads per process, M=memory per process
- numa01
  - M=3GiB P=2, T=nr_cpus/2, all threads share all process memory
- numa01_THREAD_ALLOC
  - M=3GiB, P=2, T=nr_cpus/2, all threads use per-thread local memory
- numa02
  - M=1GiB, P=1, T=nr_cpus, all threads use per-thread local memory
- numa02_SMT
  - M=1GiB, P=1, T=nr_cpus/2, all threads use per-thread local memory
    - The kernel to get this right must not use more than one HT thread per core, and in turn it must decide to split the load over two NUMA nodes even if the load would fit in a single NUMA node
    - Testing with T=nr_cpus/4 and smaller T values, would also be interesting, but if the kernel behaves well with T=nr_cpus/2 there's a good chance it'll behave sanely with more than half of the CPUs idle too
- More will be added...

numa01 (startup)



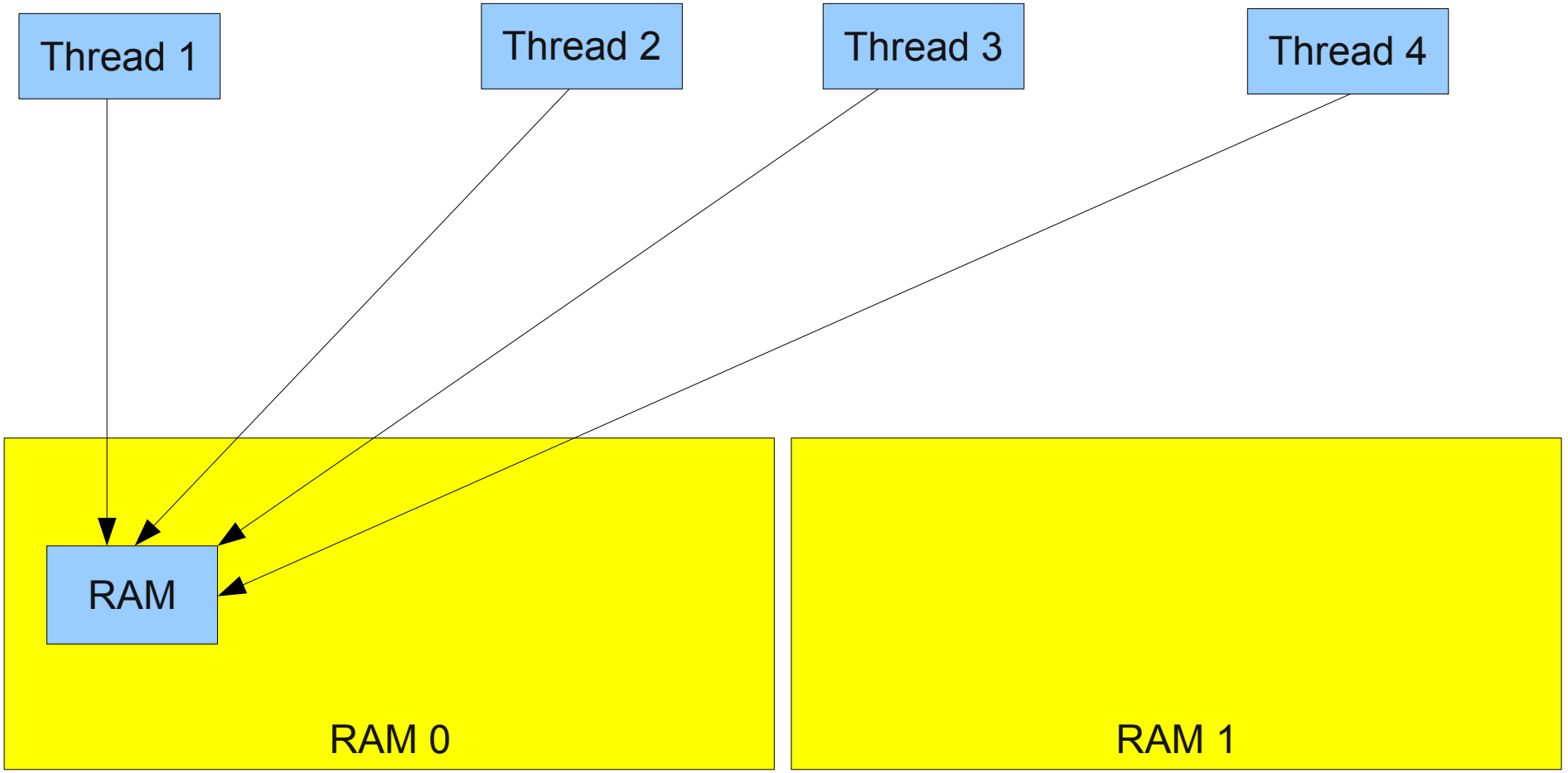
redhat

numa01 (converged)



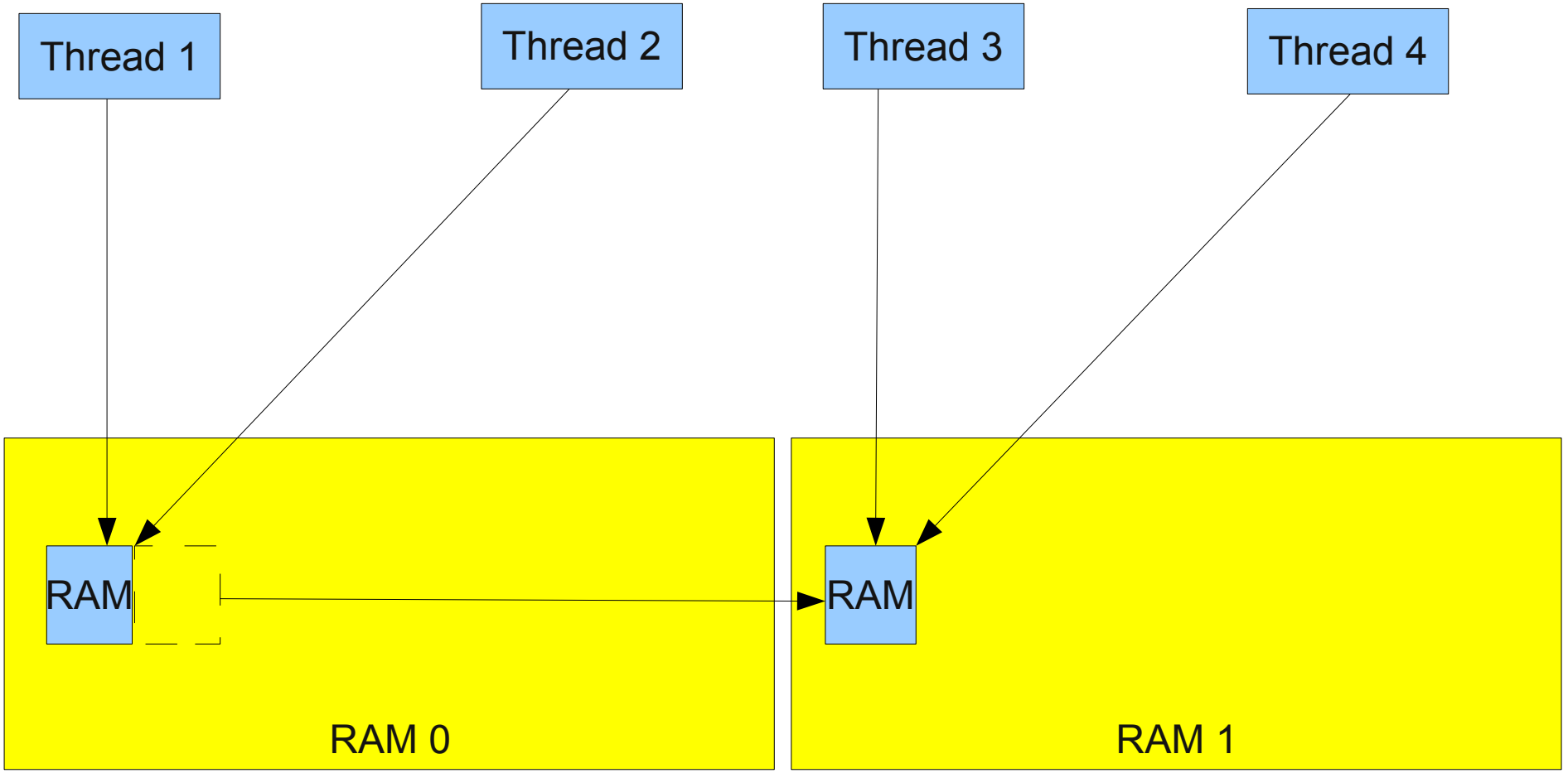
redhat

numa02 (startup)



redhat

numa02 (converged)



redhat

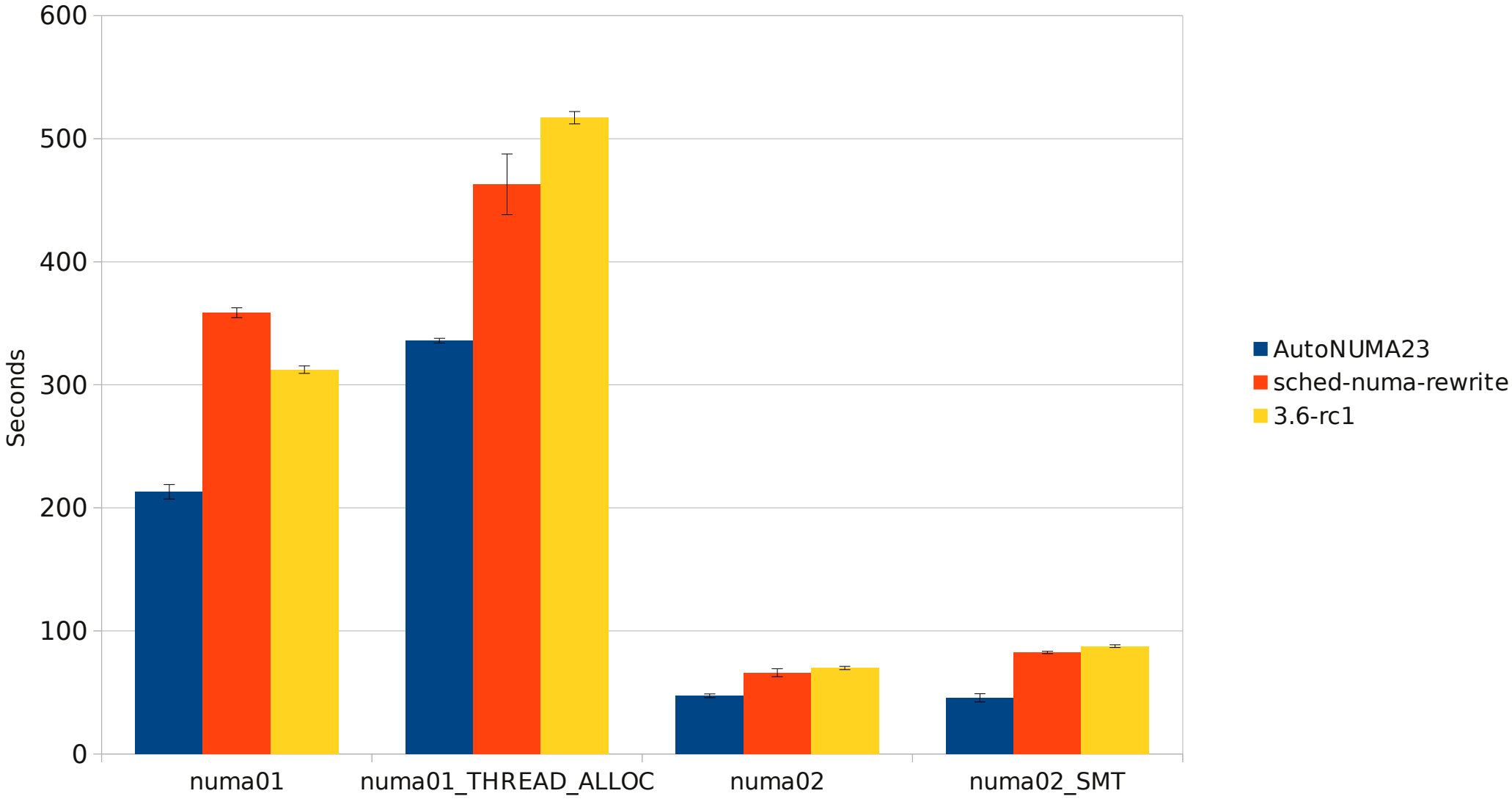
autonuma-benchmark

```
$ git clone git://gitorious.org/autonuma-benchmark/autonuma-benchmark.git
$ cd autonuma-benchmark
$ sudo ./start_bench.sh -s -t
```


redhat.

4 runs, average and stdev

Lower is better

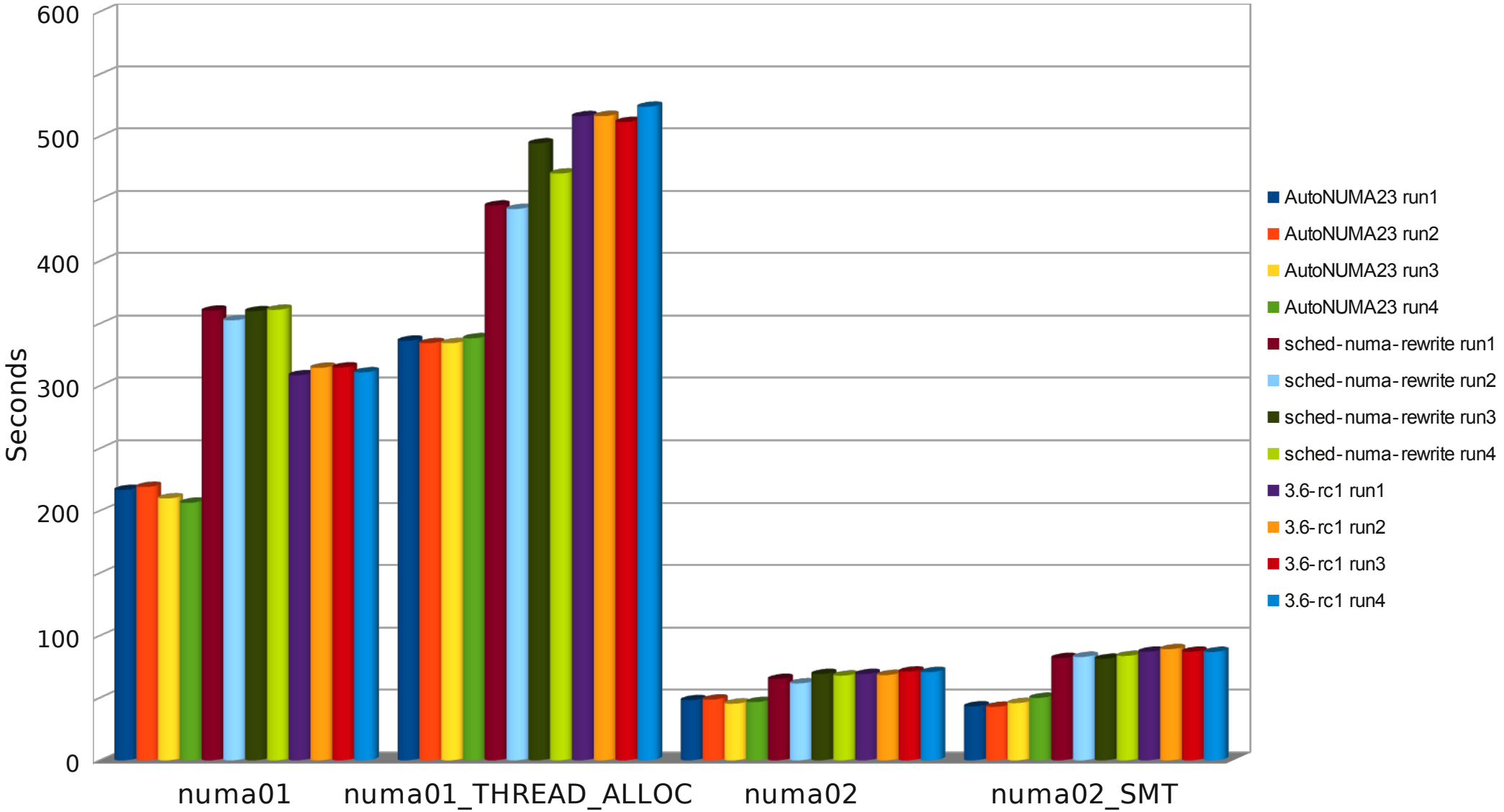


2 nodes

redhat.

4 runs of each test, all results

Lower is better

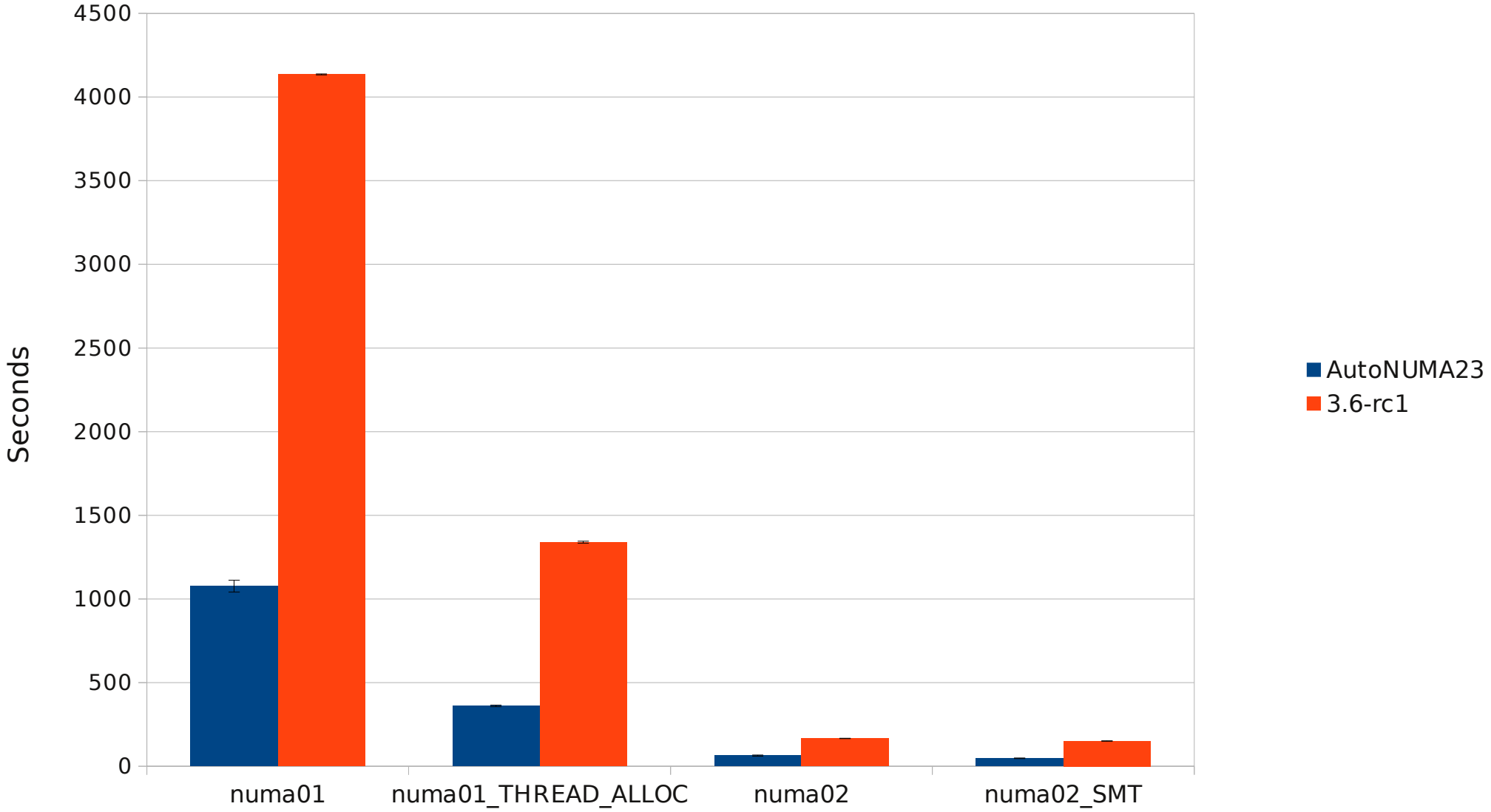


2 nodes

redhat

4 runs, average and stdev

Lower is better

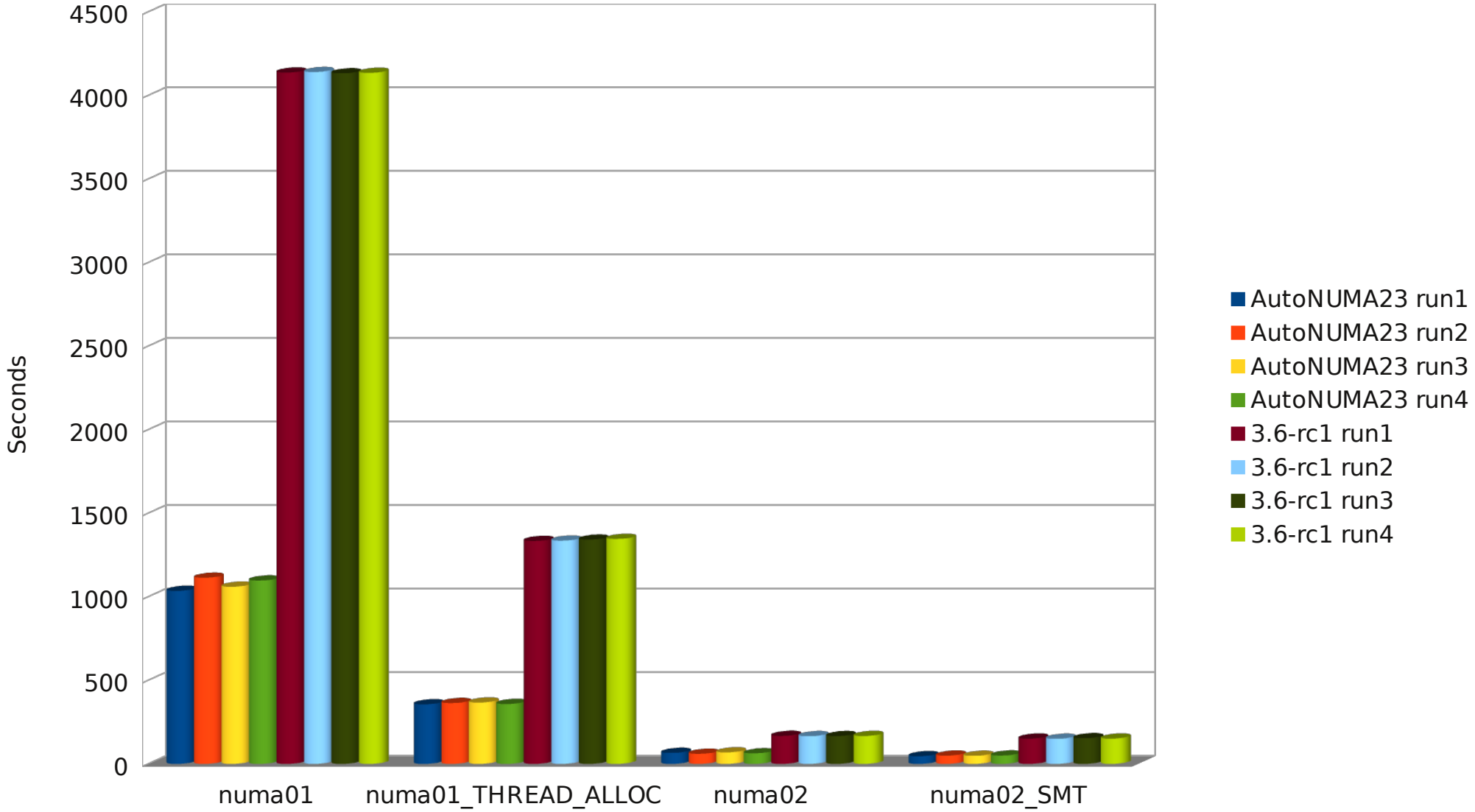


8 nodes

redhat.

4 runs of each test, all results

Lower is better



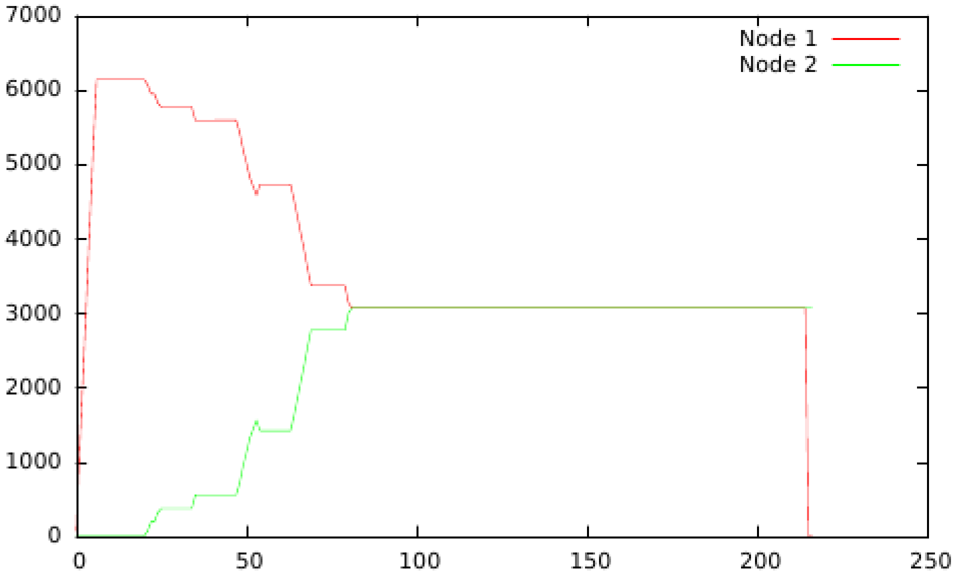
8 nodes

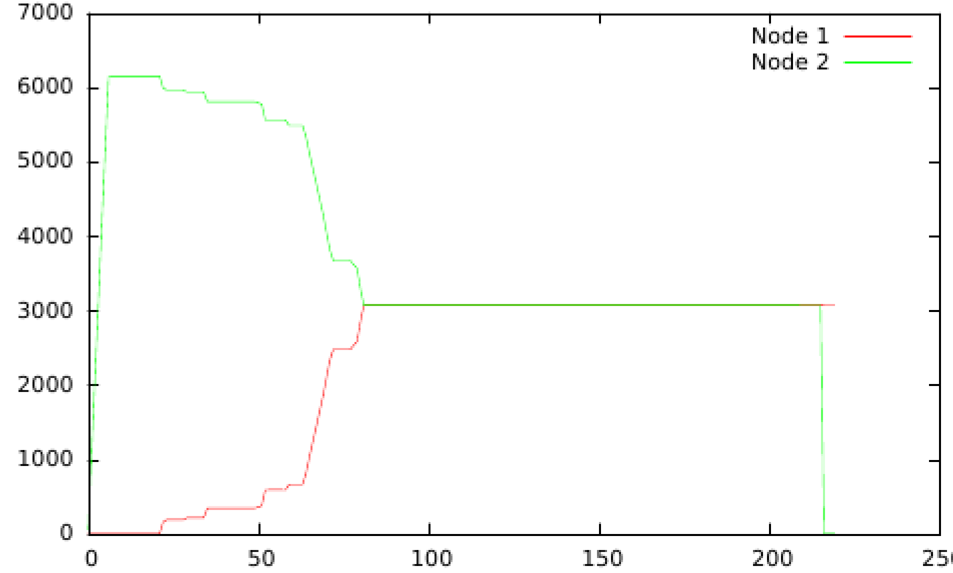
redhat

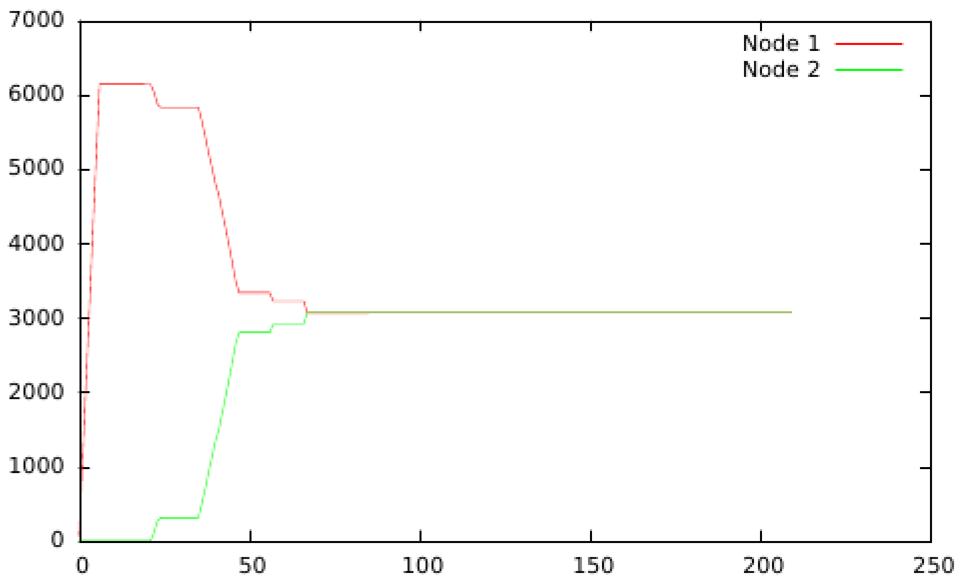
Convergence charts

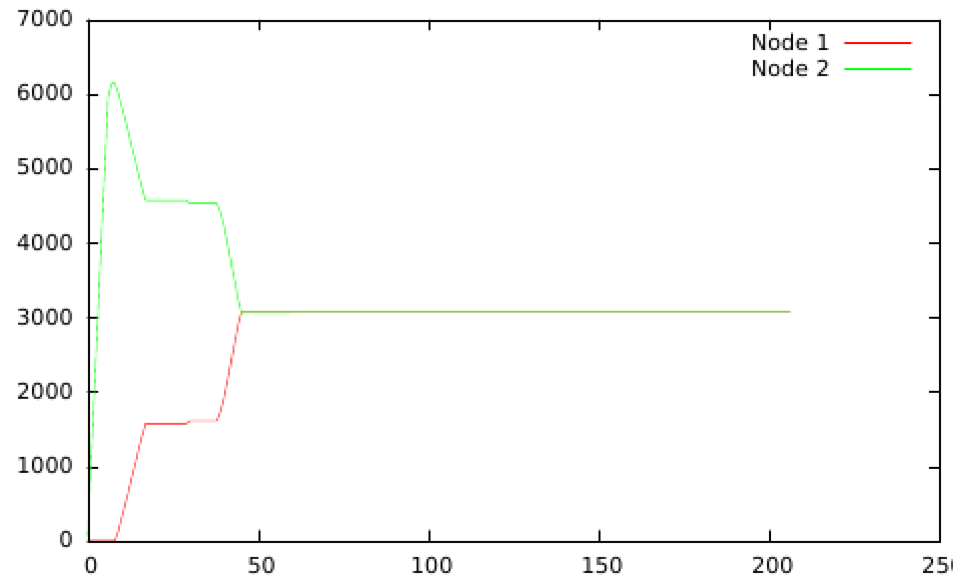
- The AutoNUMA-benchmark produces a chart for each test run (in pdf format).
  - In each chart there is one two dimensional line per NUMA node (node1, node2, etc.).
    - X=time (seconds)
    - Y=memory (MiB)
  - Each line represents how much of the test process's memory is in that NUMA node over time, for the duration of the test. Because all memory starts in one node, it illustrates how the memory migrates over time.
  - A workload converges when the memory levels are equal in all NUMA nodes
  - Note: with nr_nodes > 2, numa01 may not fully converge because half of the cpus will thrash on the memory of half of the nodes, but it should get close enough
  - In the future we plan to add a new numa01 "PER_NODE" test with P=nr_nodes and T=nr_cpus/nr_nodes

AutoNUMA23 numa01



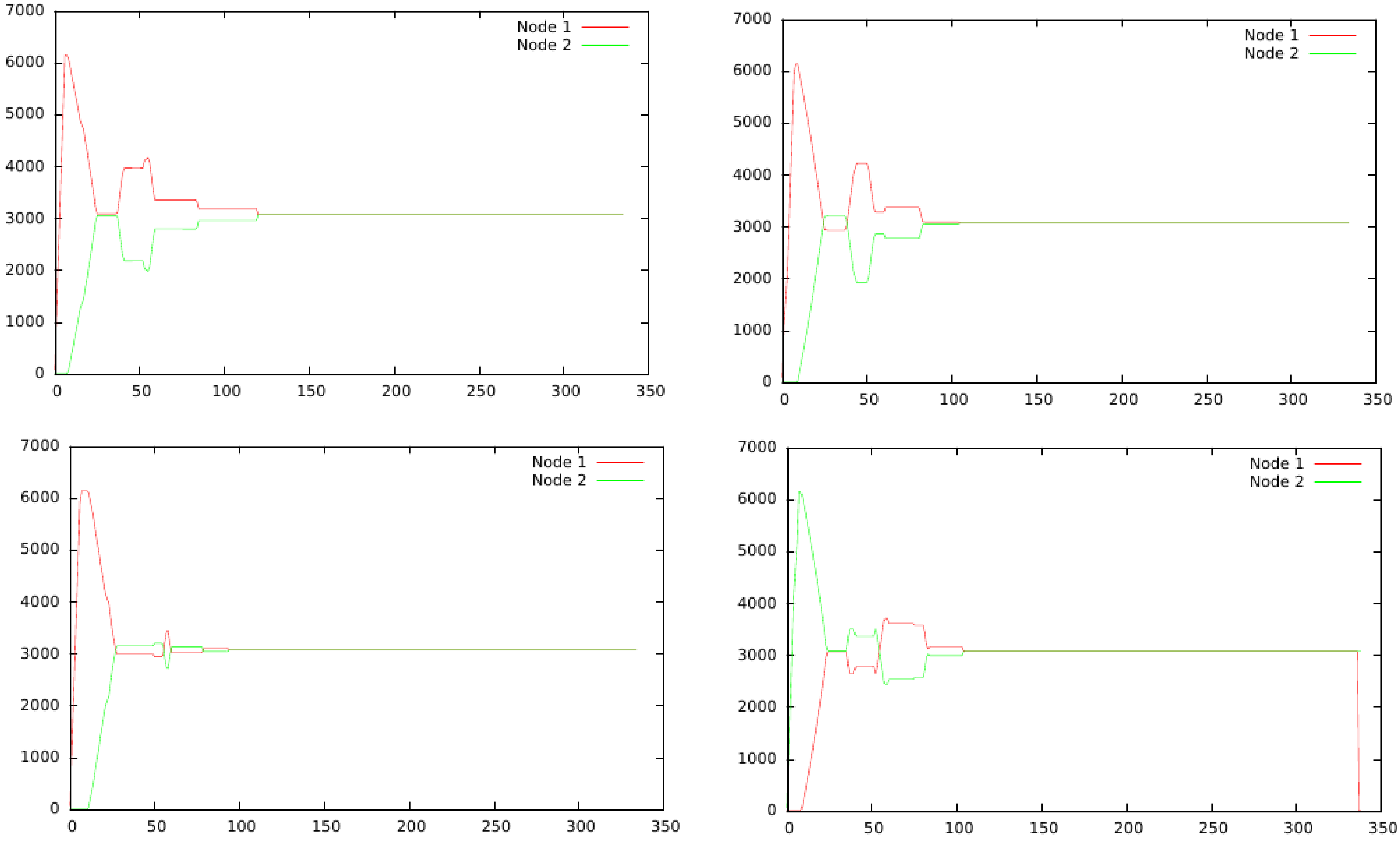






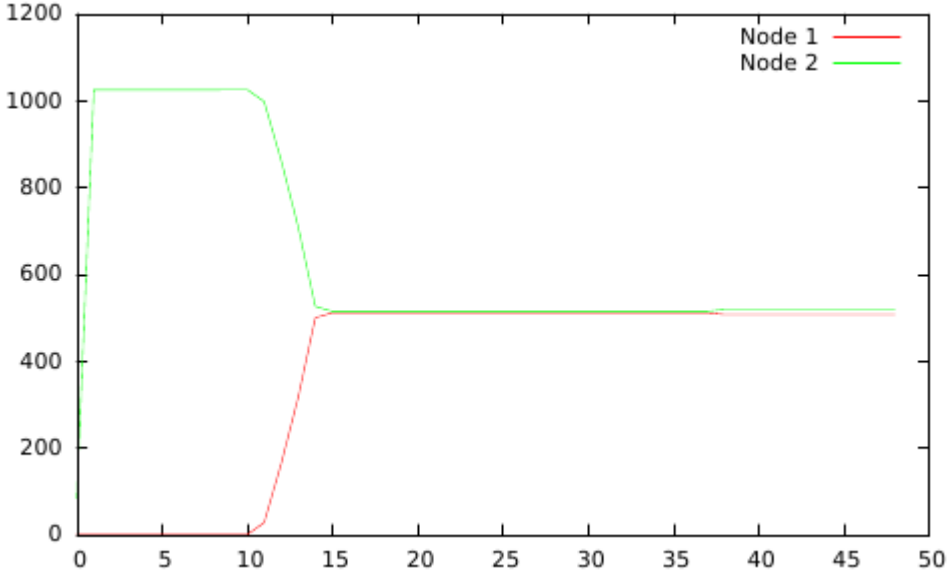
2 nodes

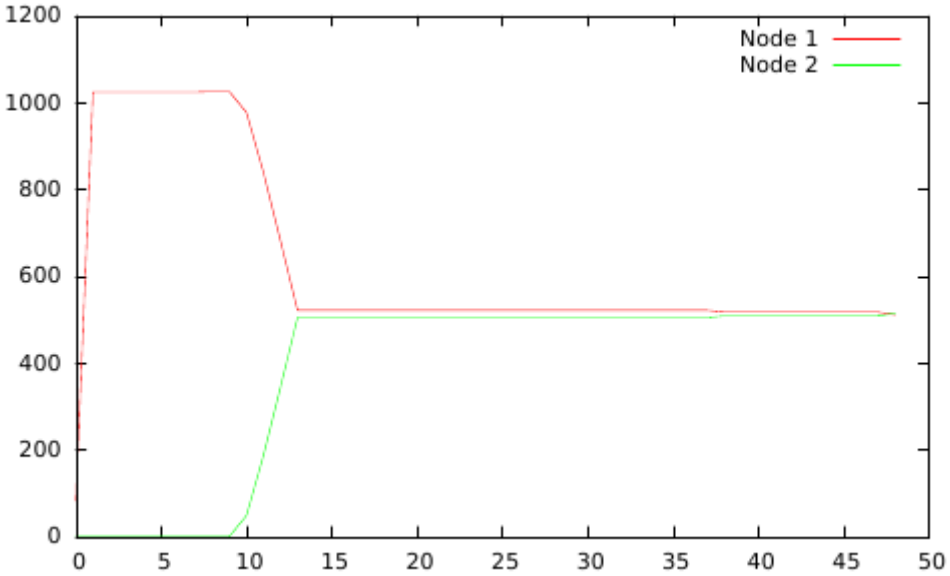
AutoNUMA23 numa01_THREAD..

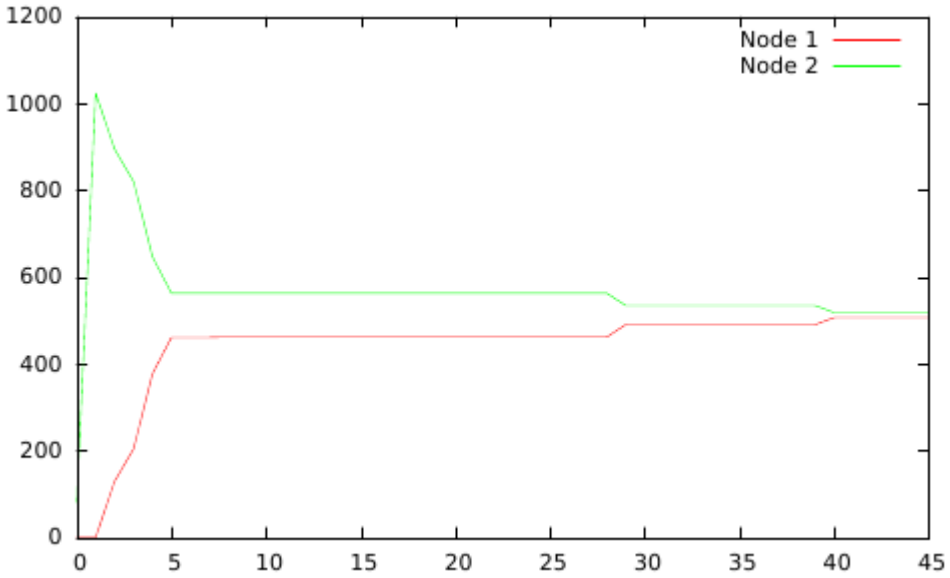


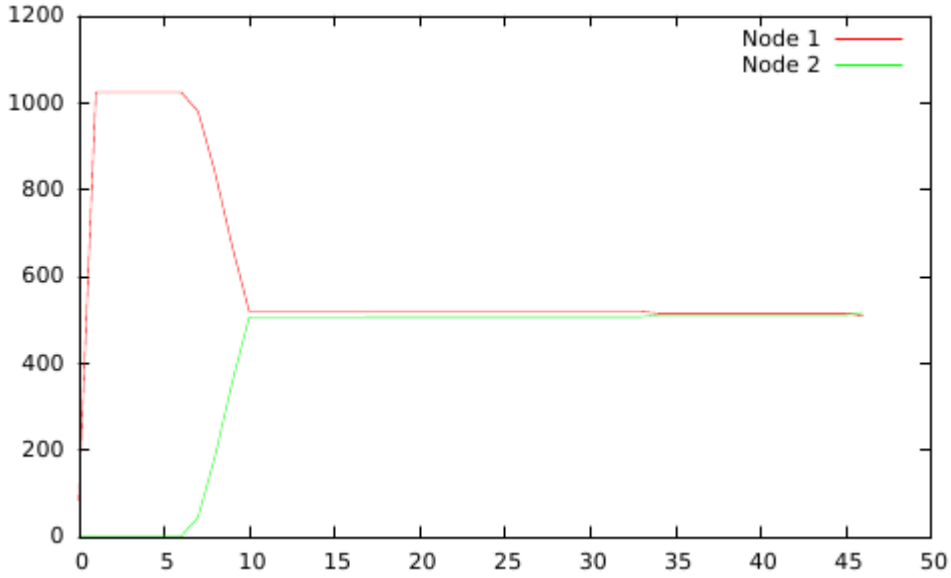
2 nodes

AutoNUMA23 numa02





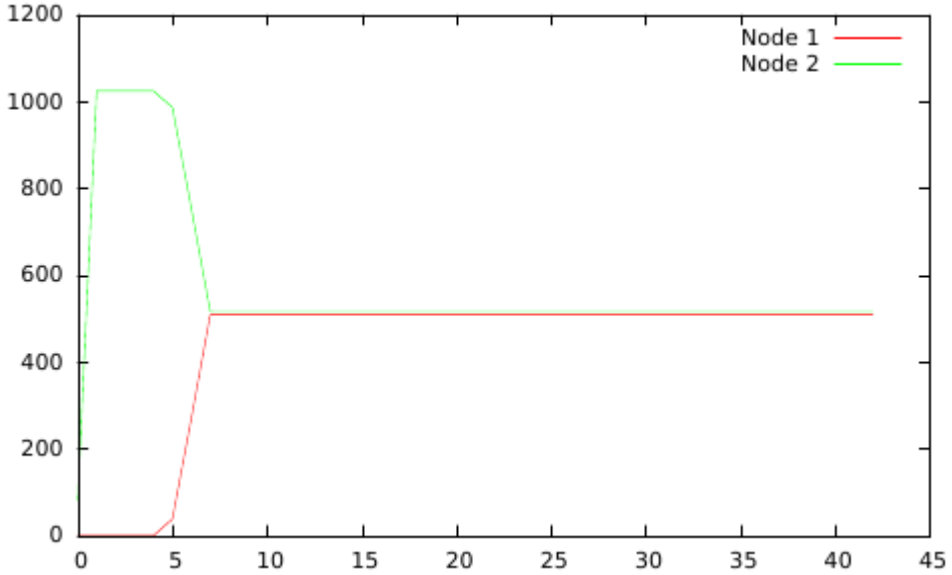


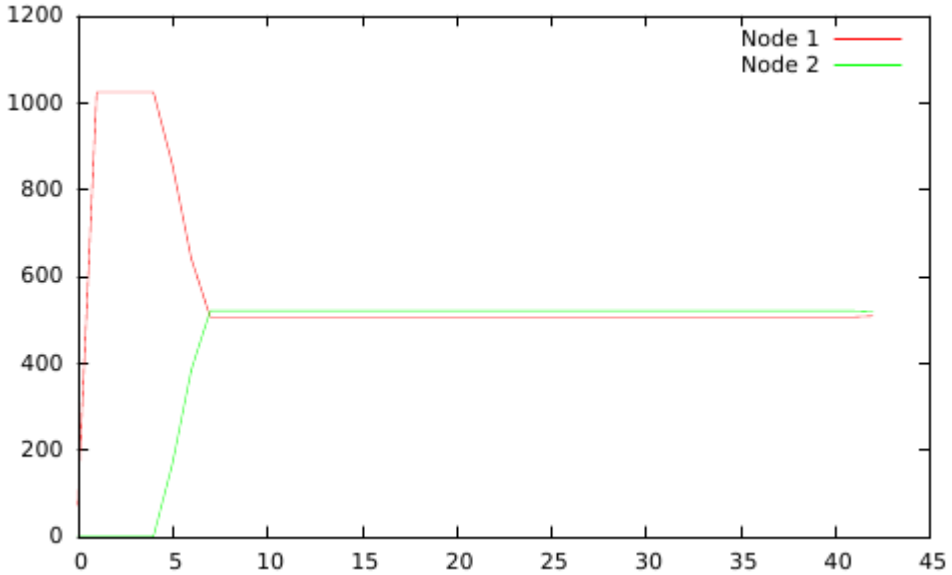


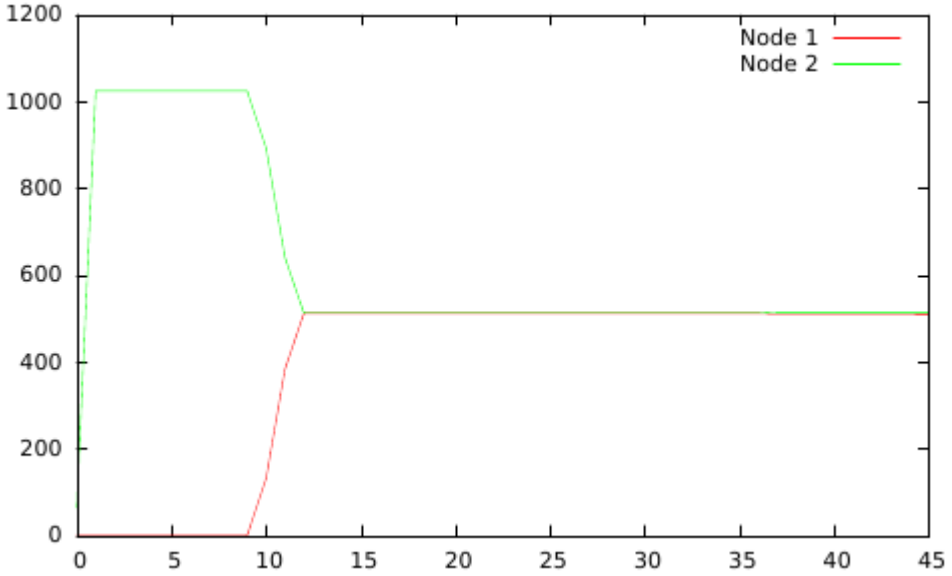
2 nodes

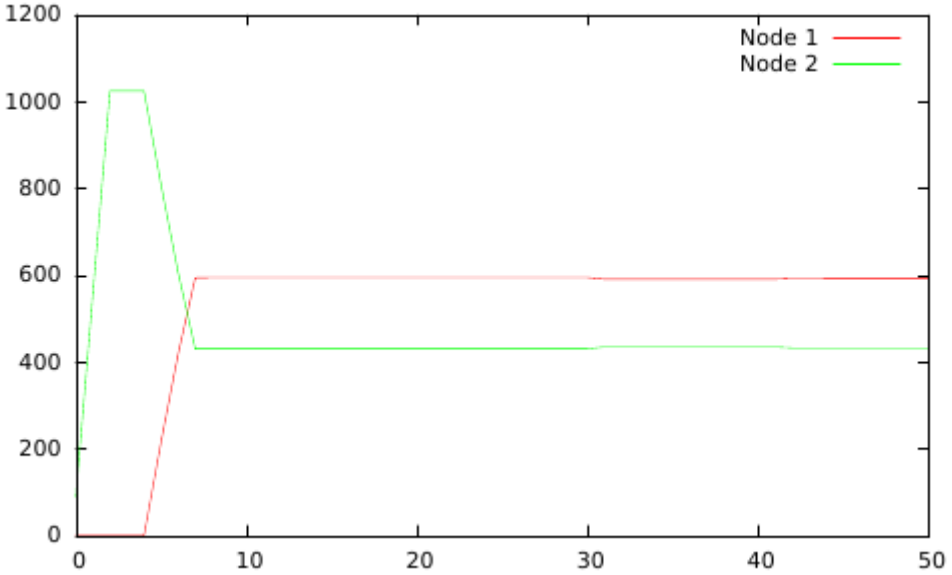
redhat.

AutoNUMA23 numa02_SMT





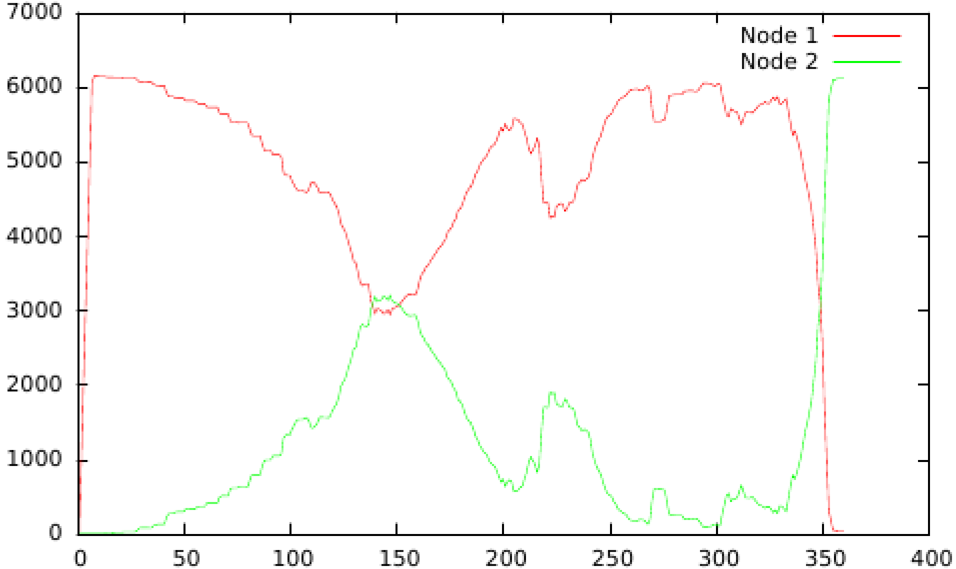


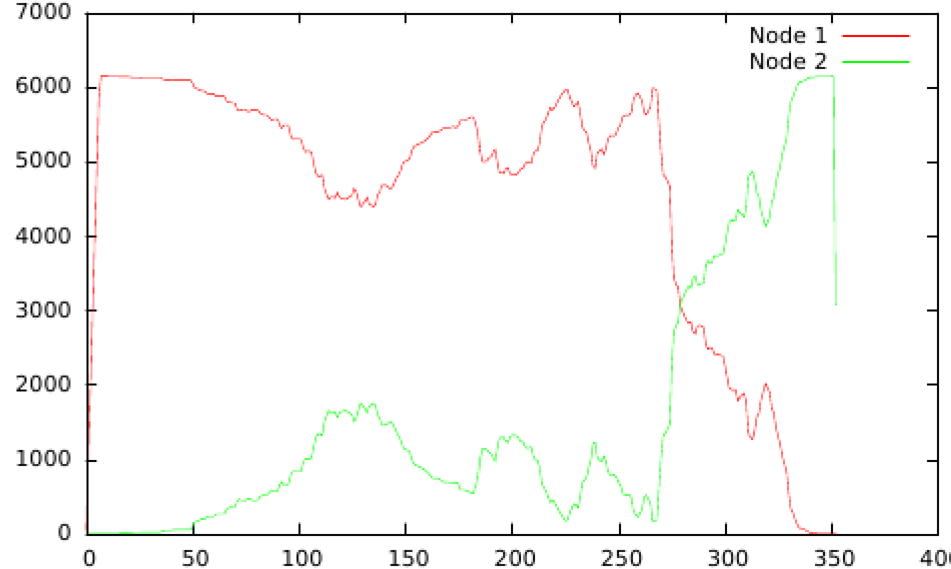


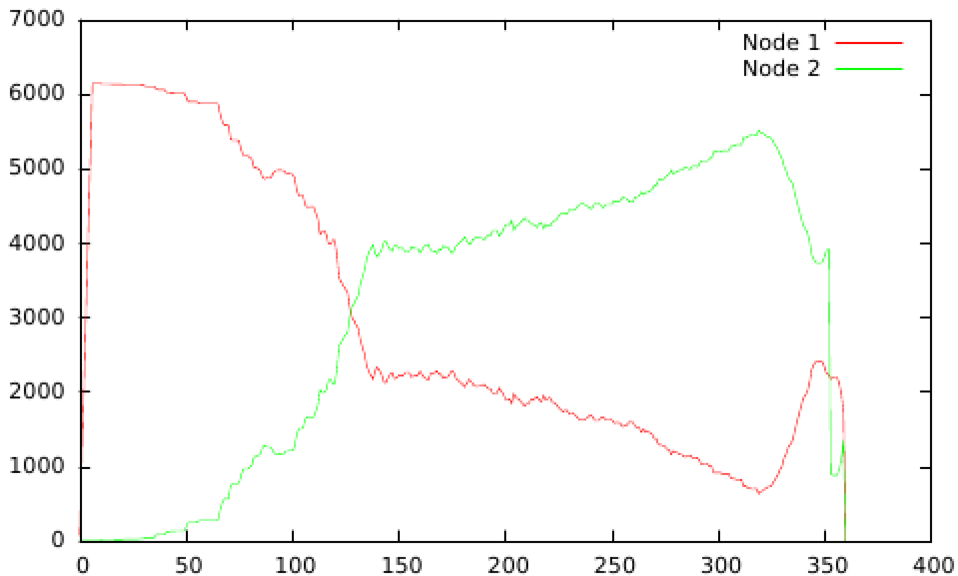
2 nodes

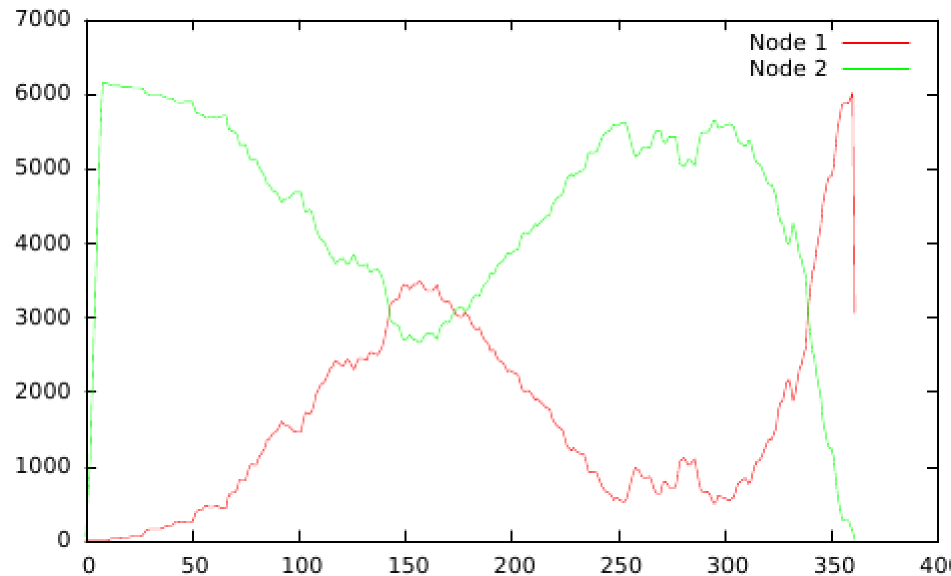
redhat.

sched-numa-rewrite numa01



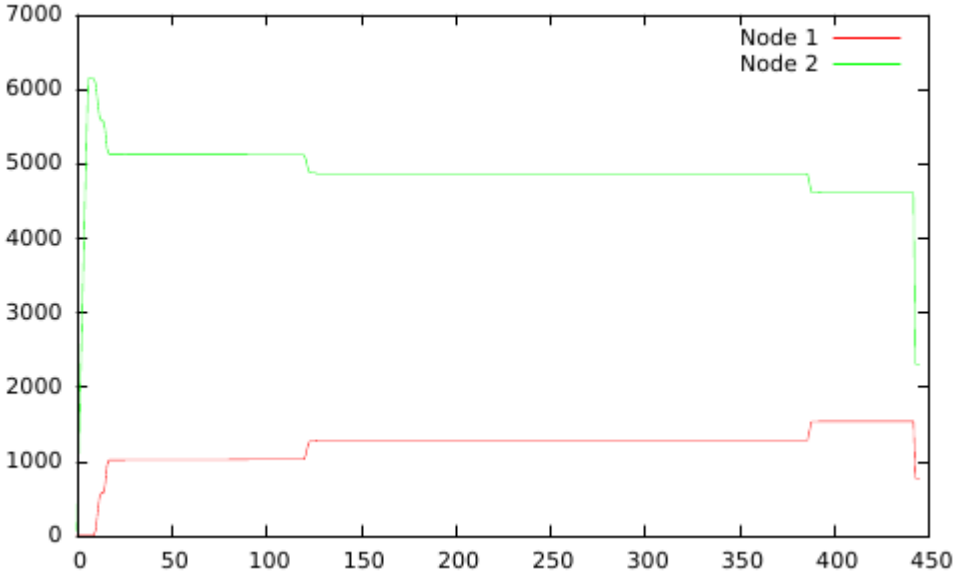


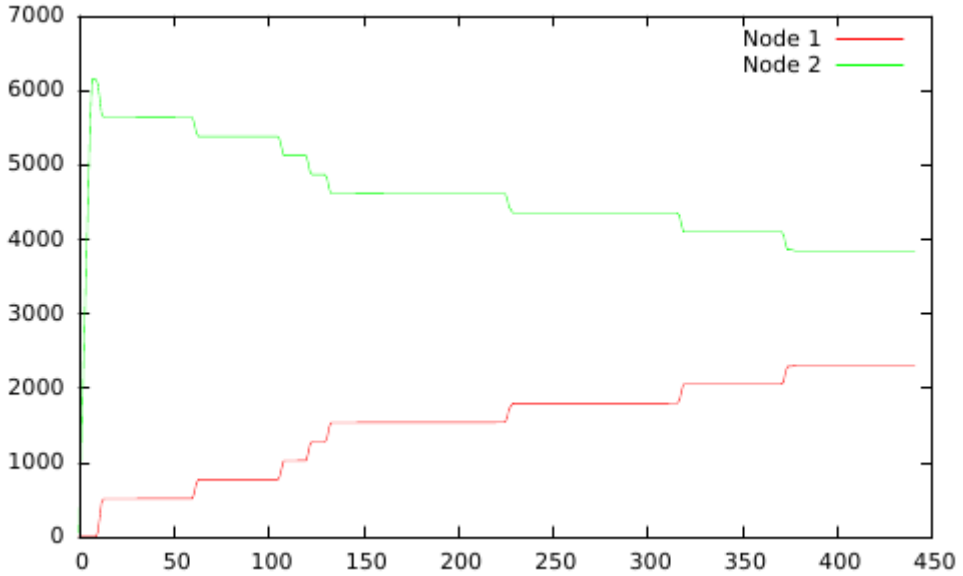


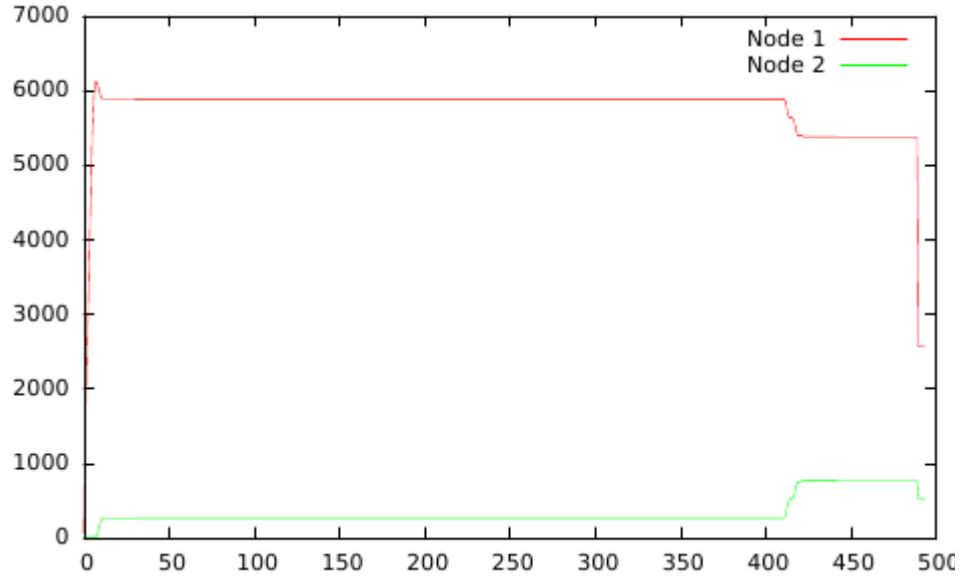


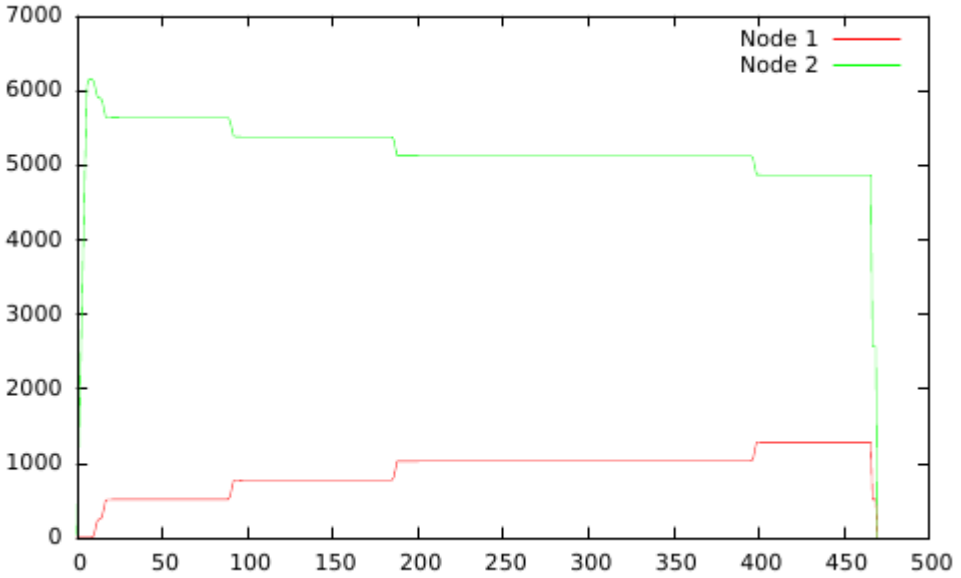
2 nodes

sched-numa-rewrite numa01_TH..





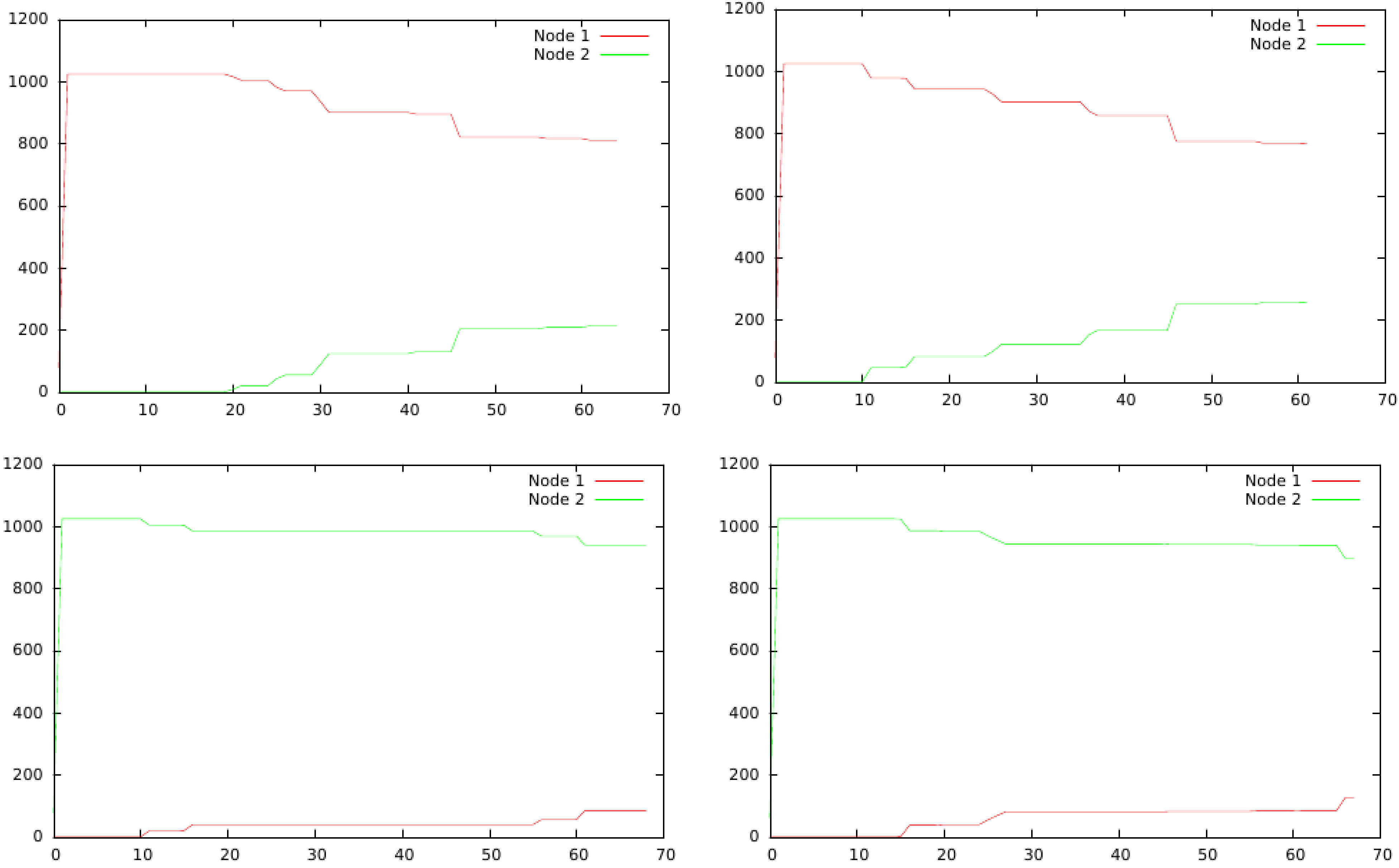




2 nodes

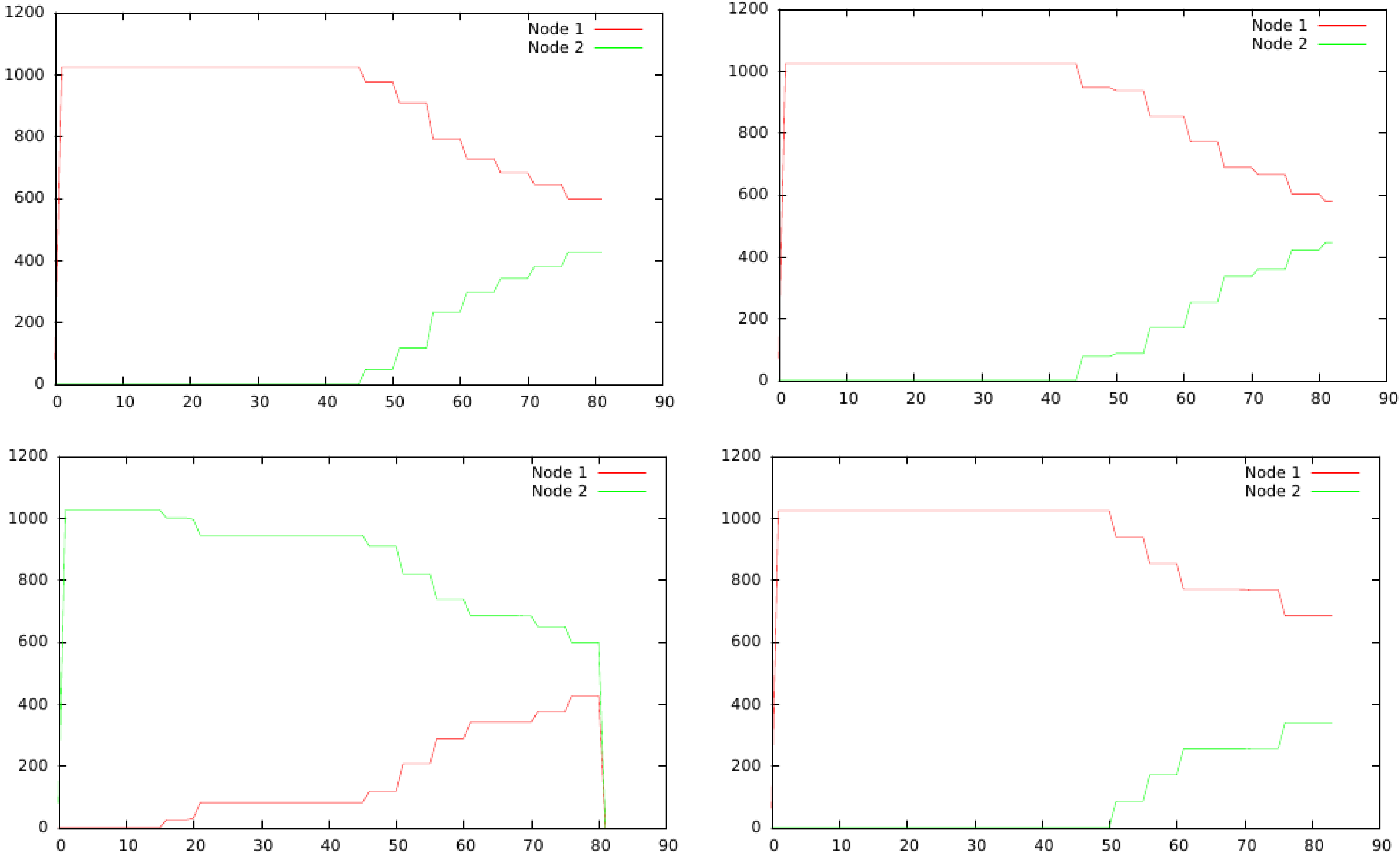
redhat.

sched-numa-rewrite numa02



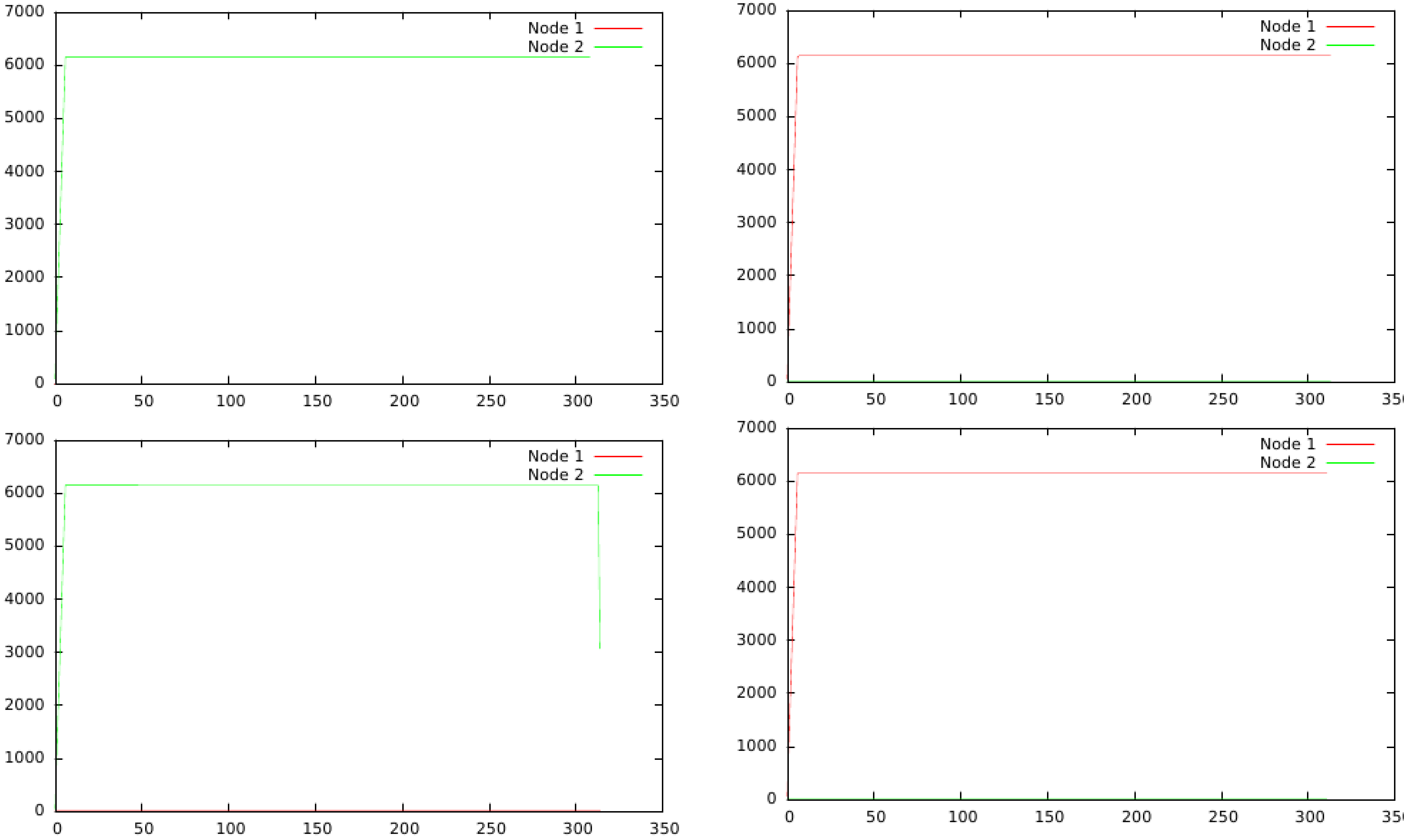
2 nodes

sched-numa-rewrite numa02_SMT



2 nodes

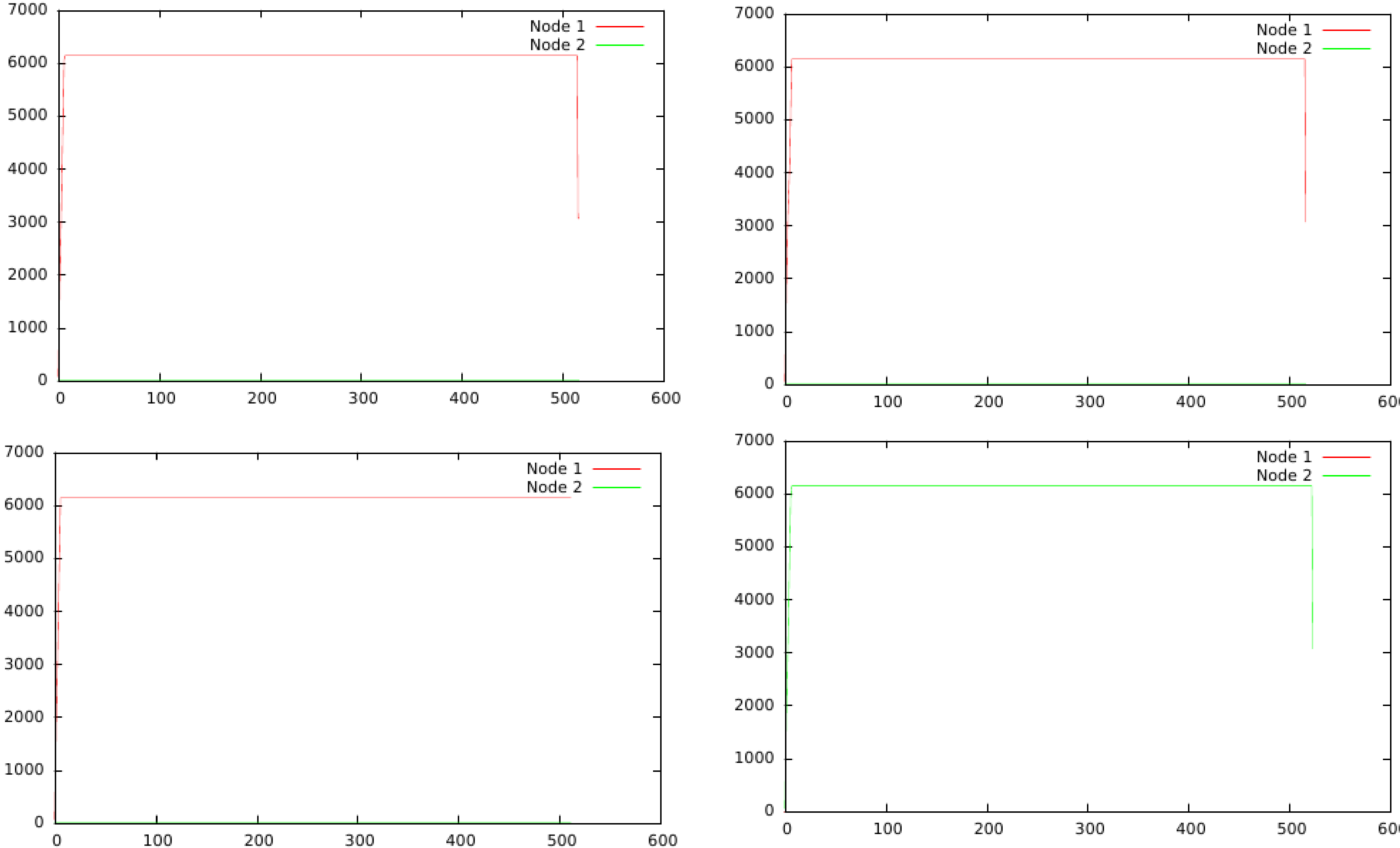
3.6-rc1 numa01



2 nodes

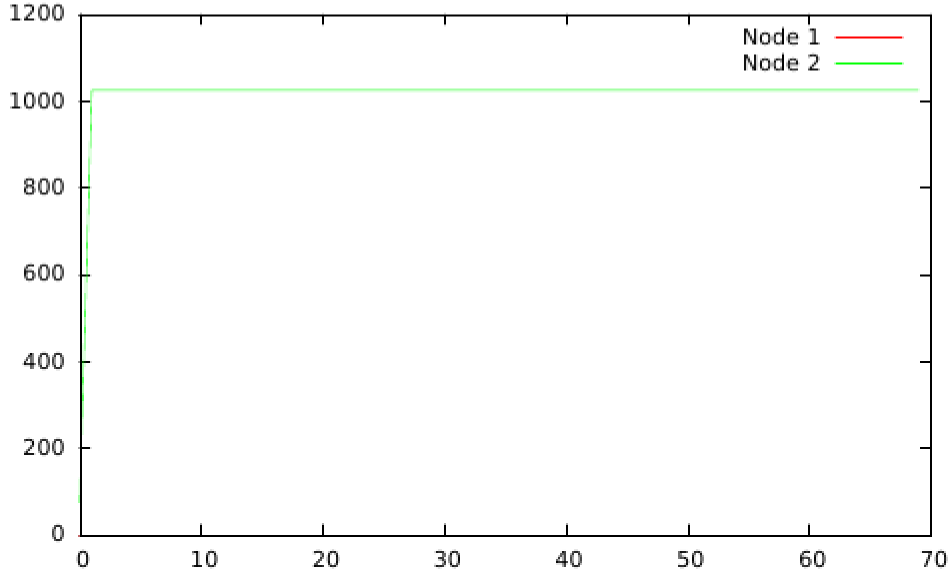
redhat

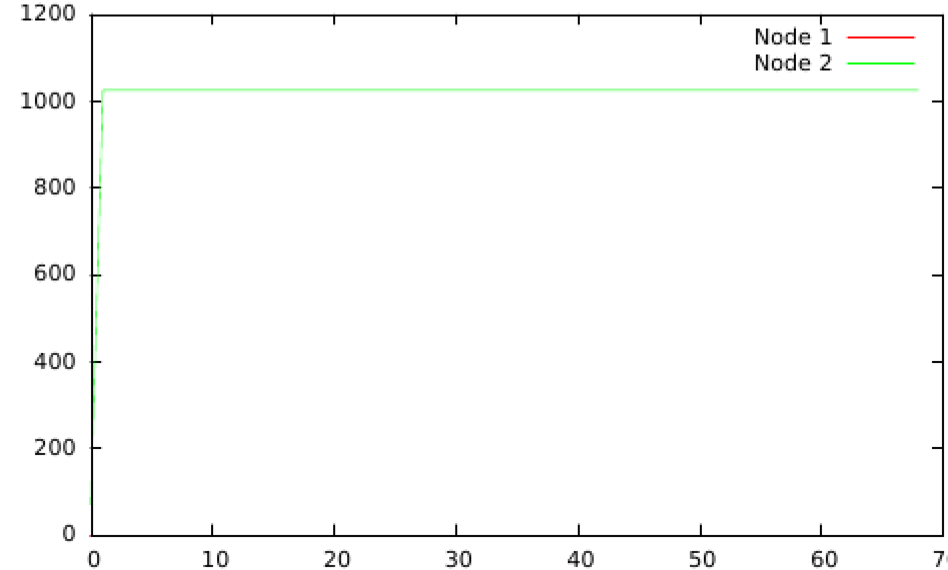
3.6-rc1 numa01_THREAD_ALLOC

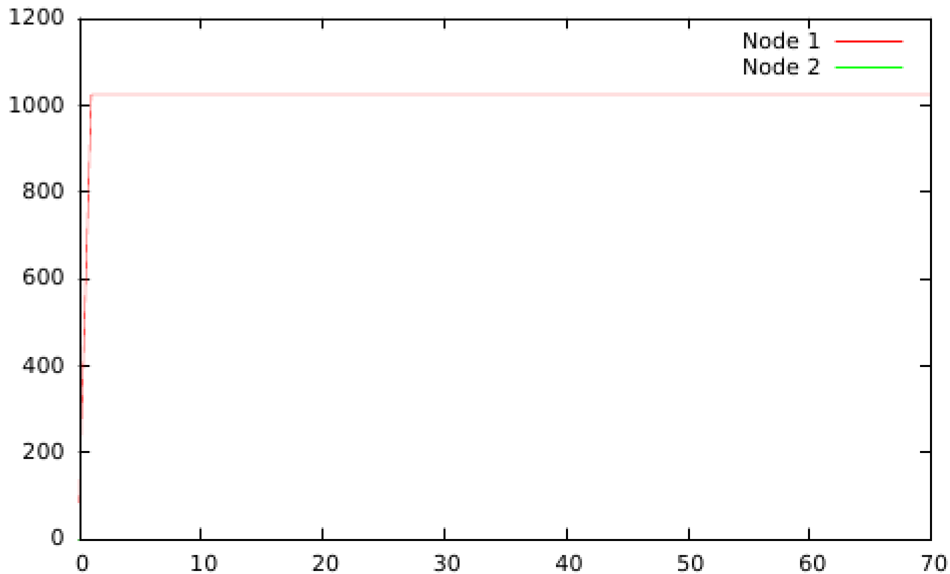


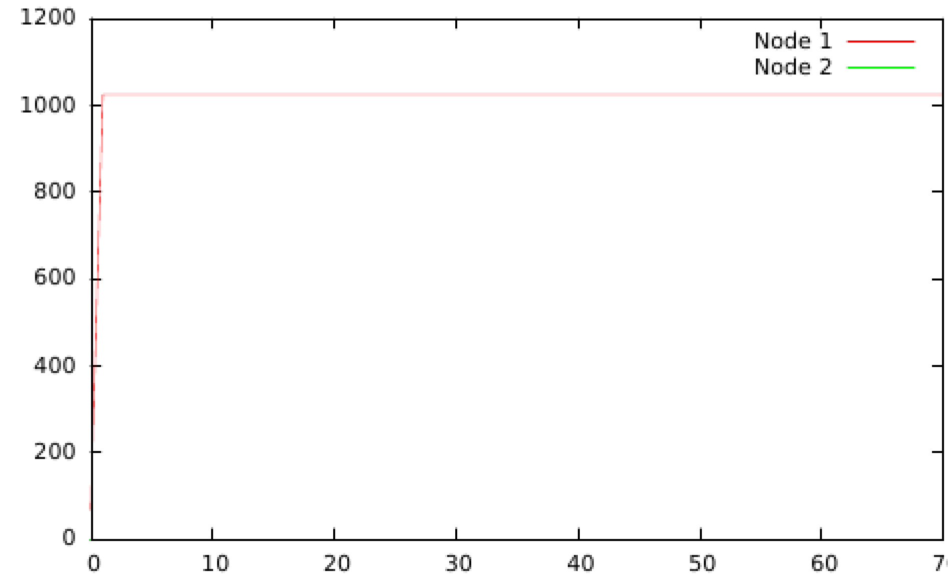
2 nodes

3.6-rc1 numa02



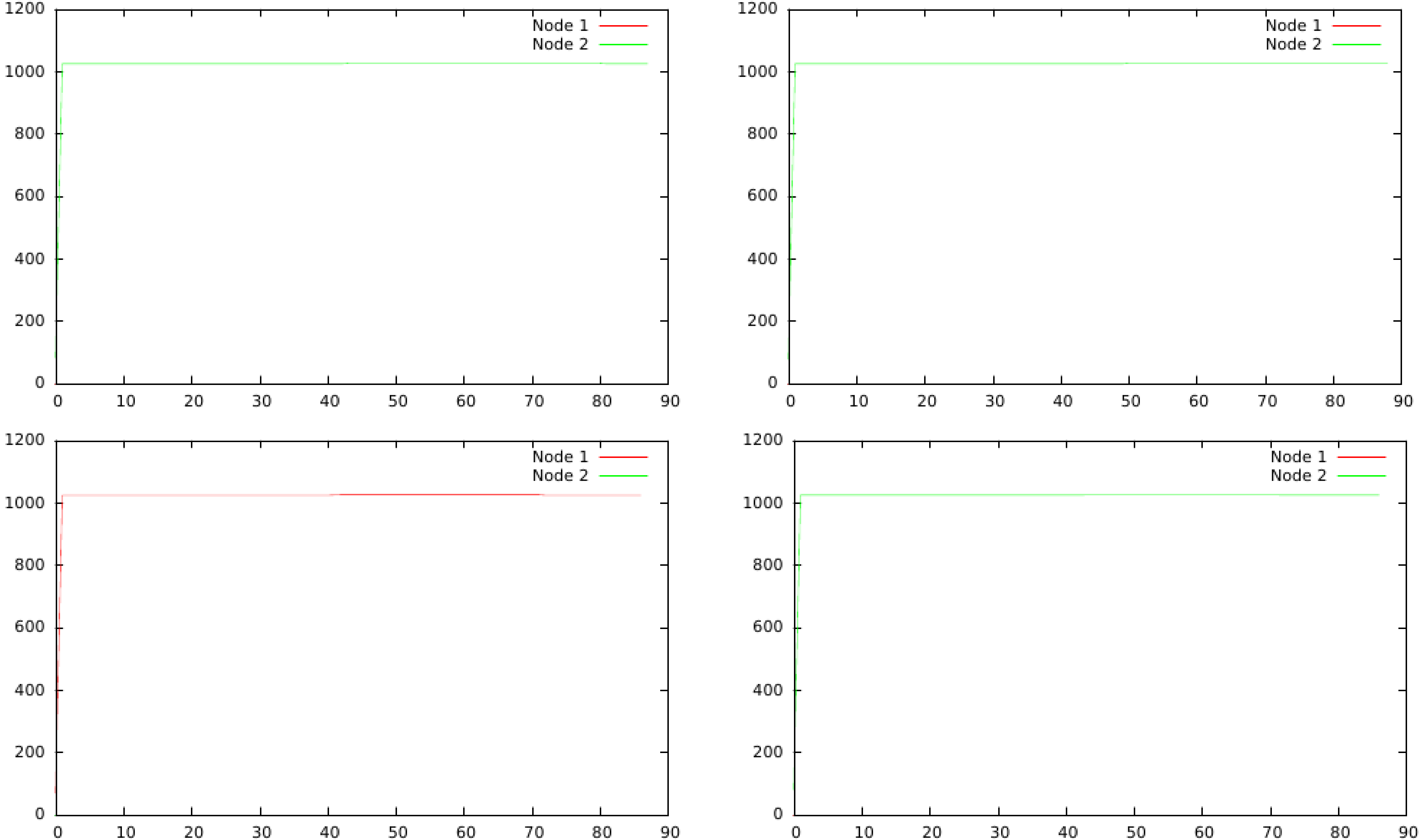






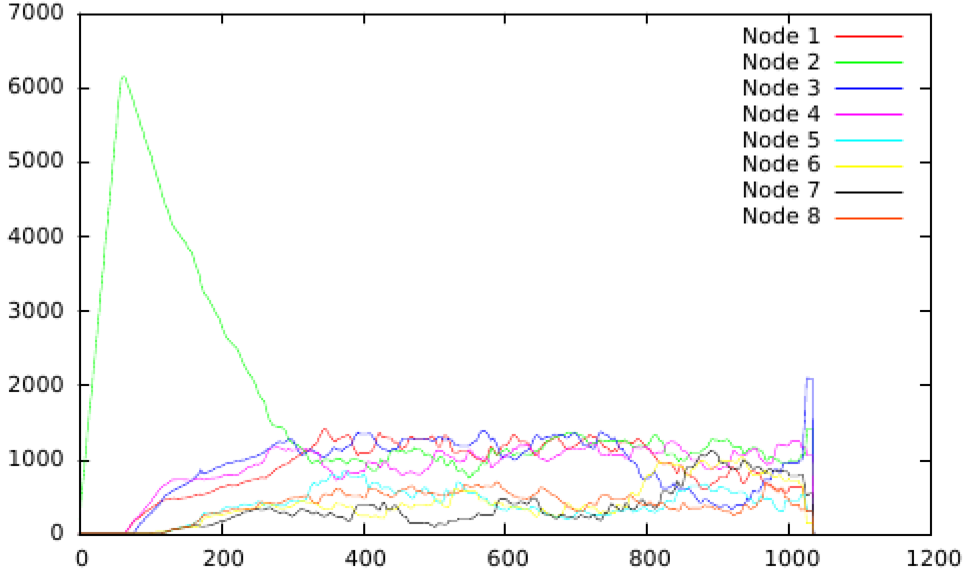
2 nodes

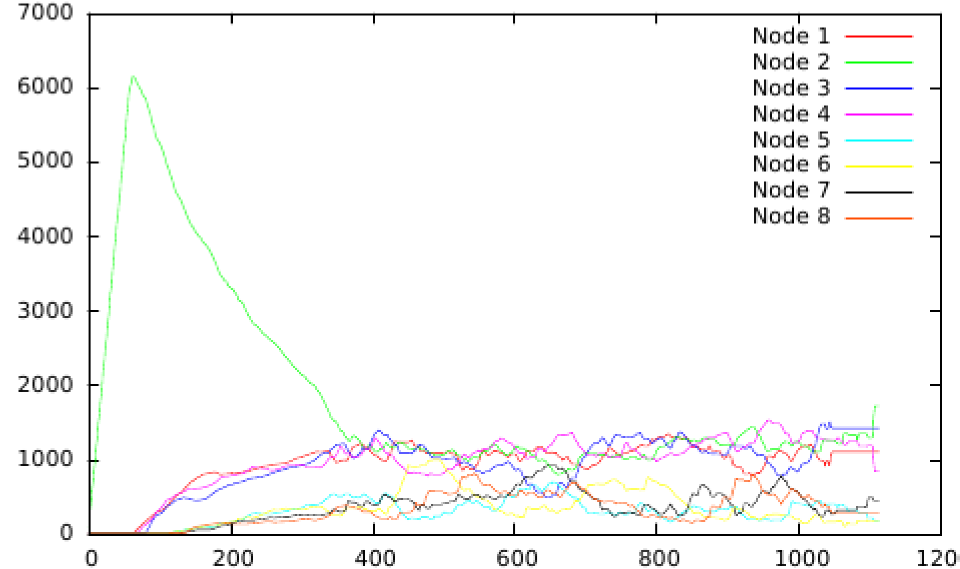
3.6-rc1 numa02_SMT

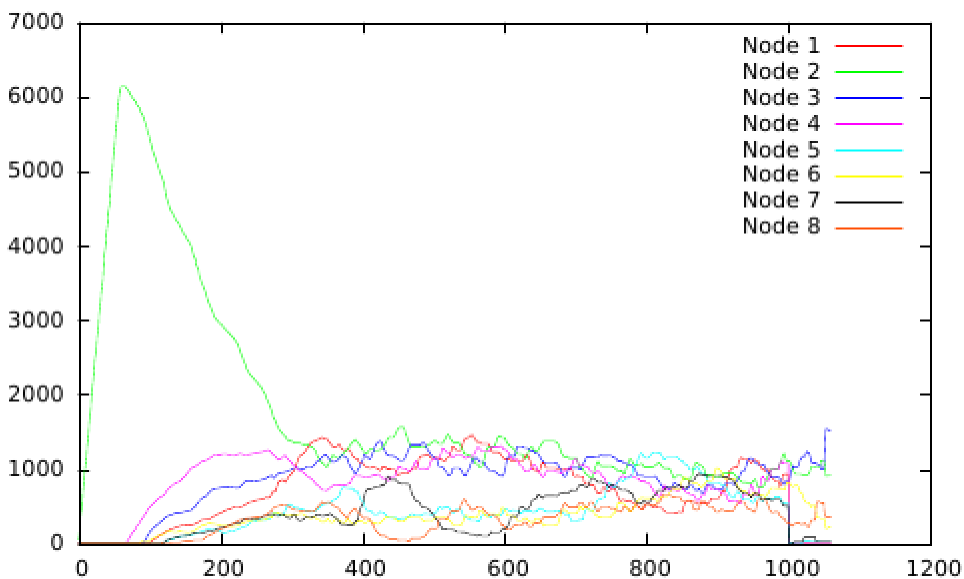


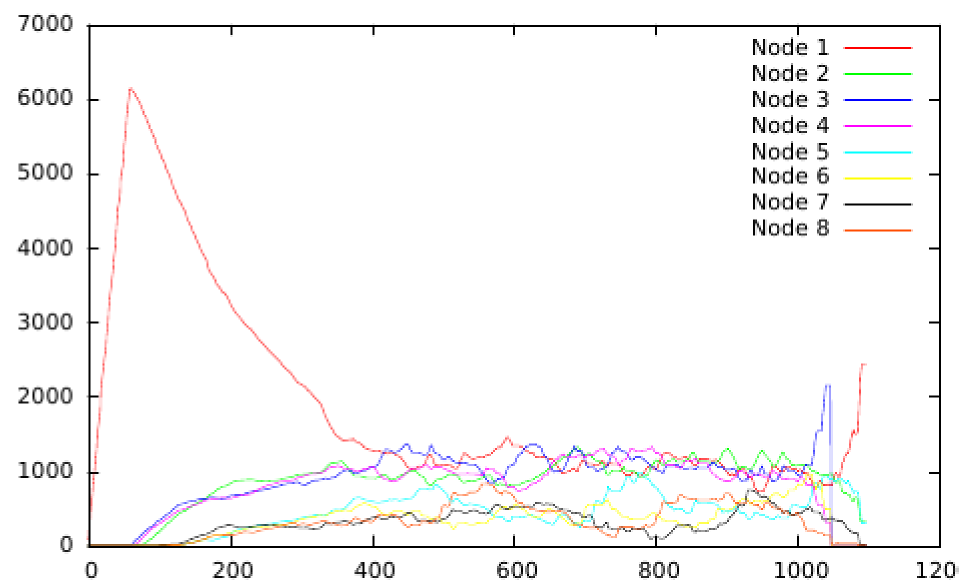
2 nodes

AutoNUMA23 numa01



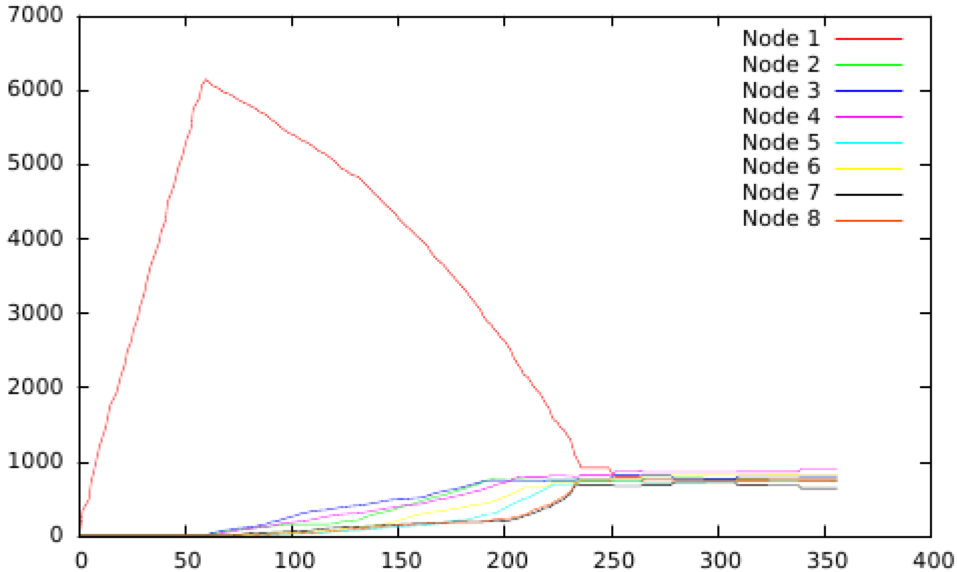


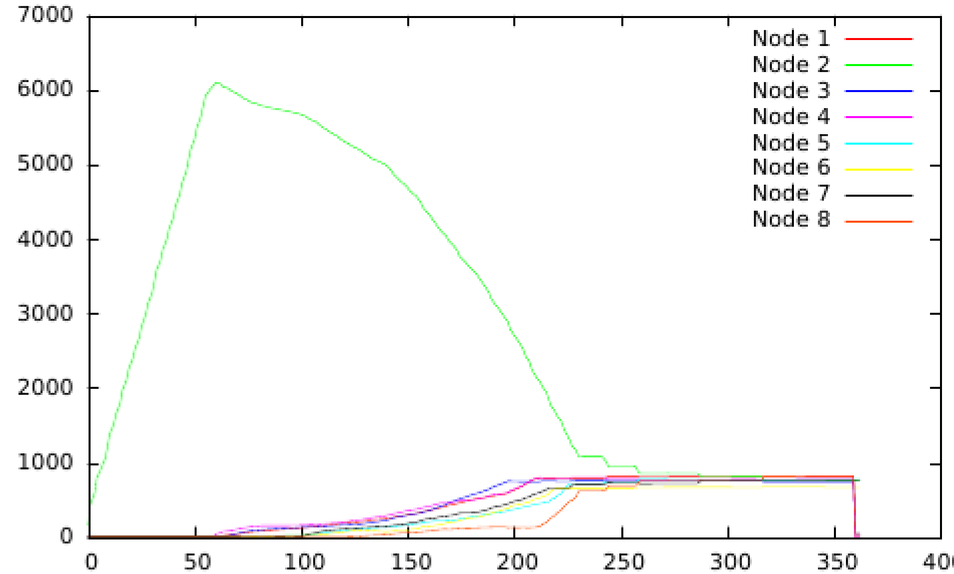


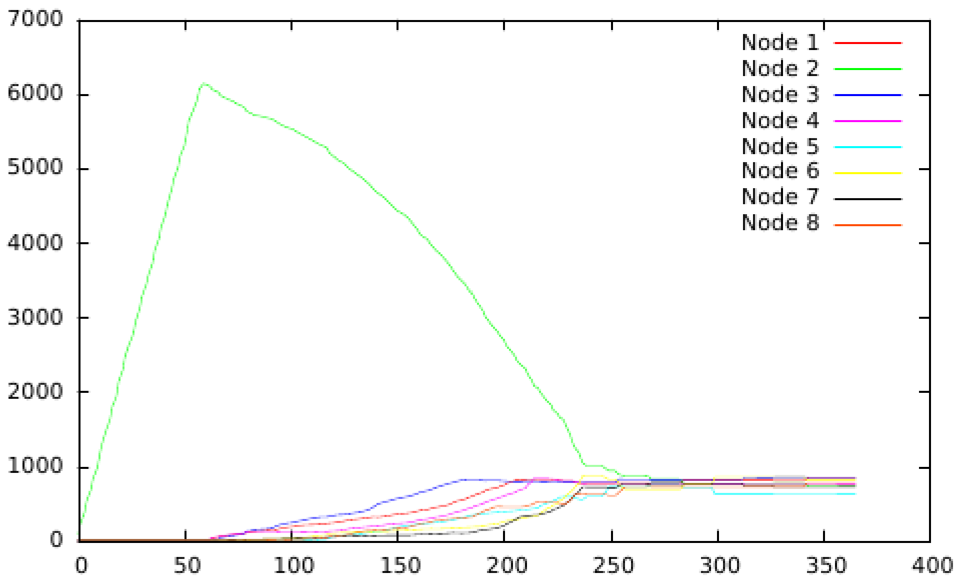


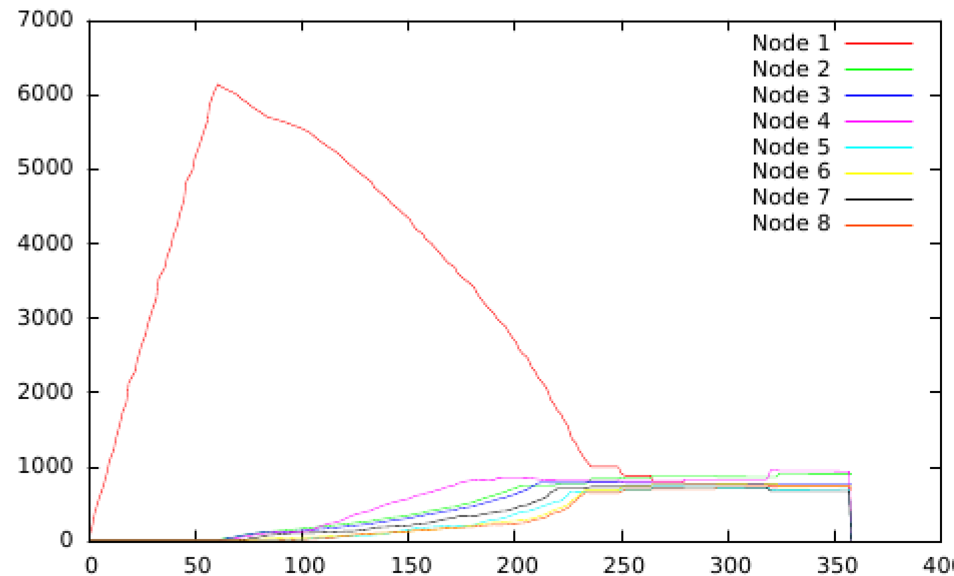
8 nodes → heavy false sharing

AutoNUMA23 numa01_THREAD..



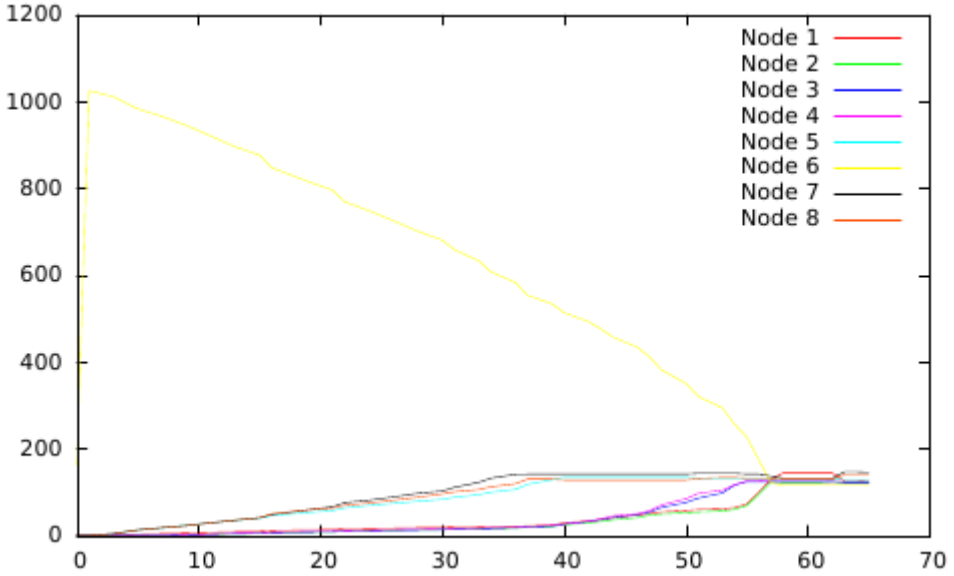


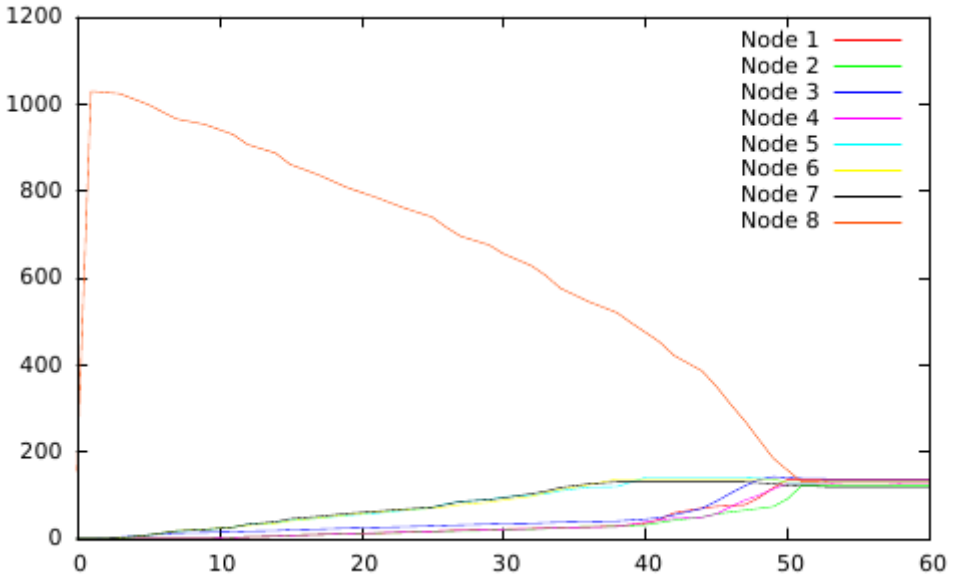


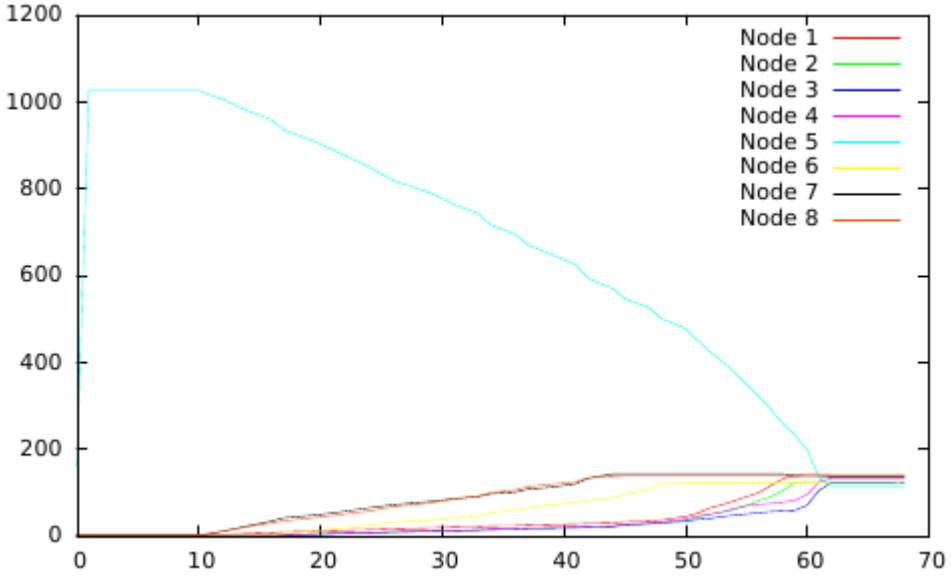


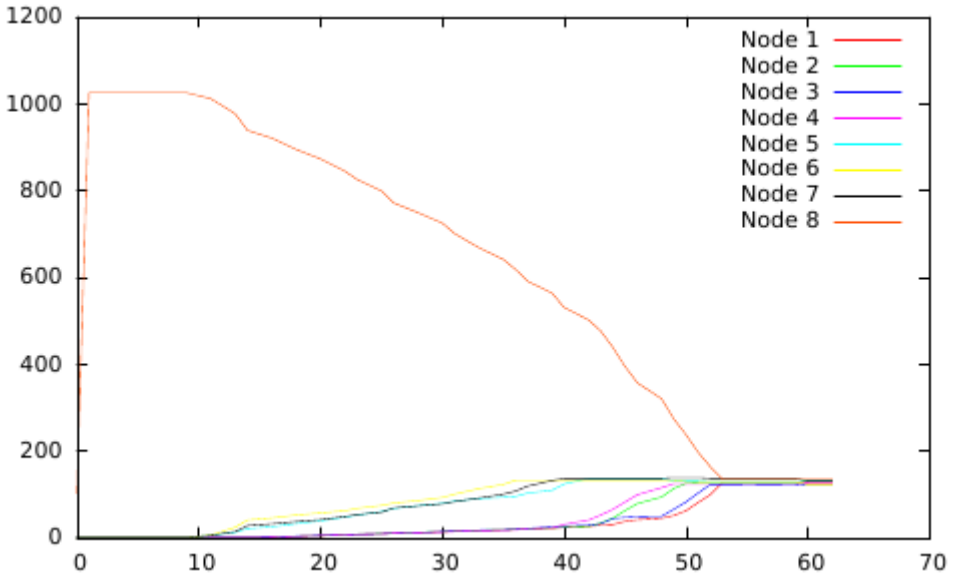
8 nodes

AutoNUMA23 numa02





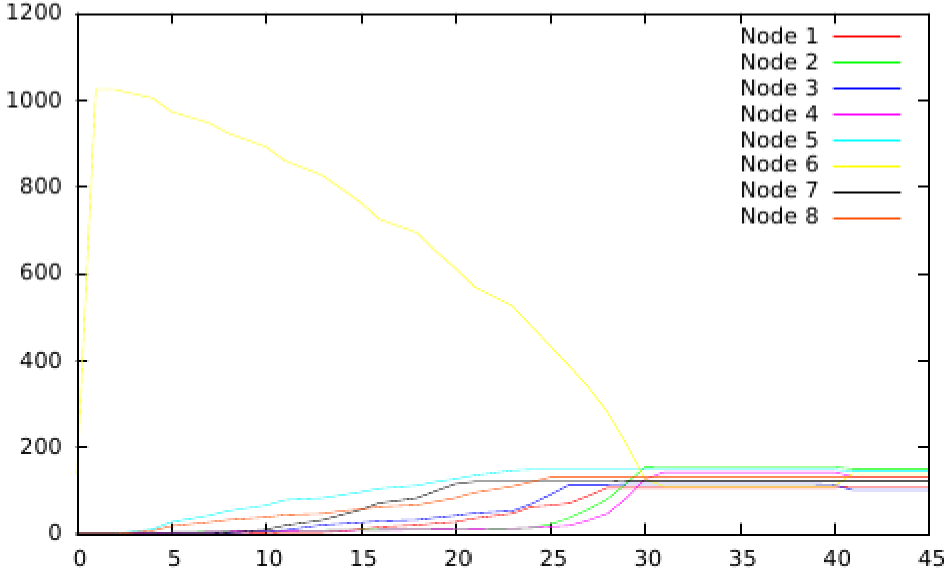


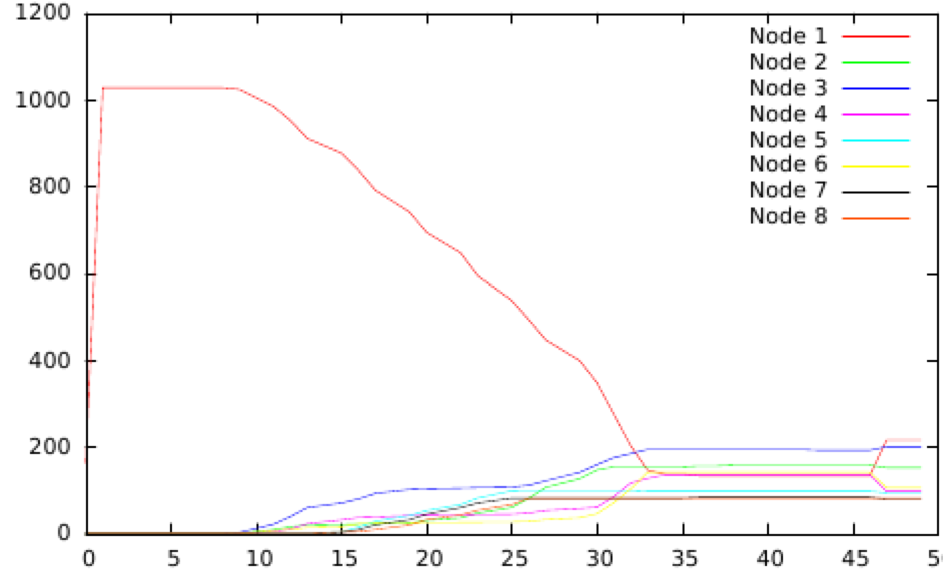


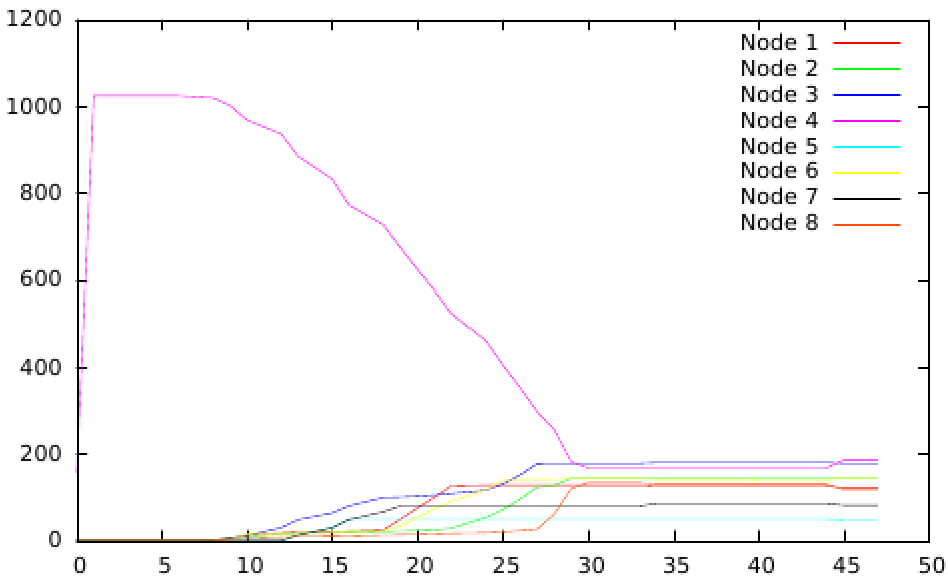
8 nodes

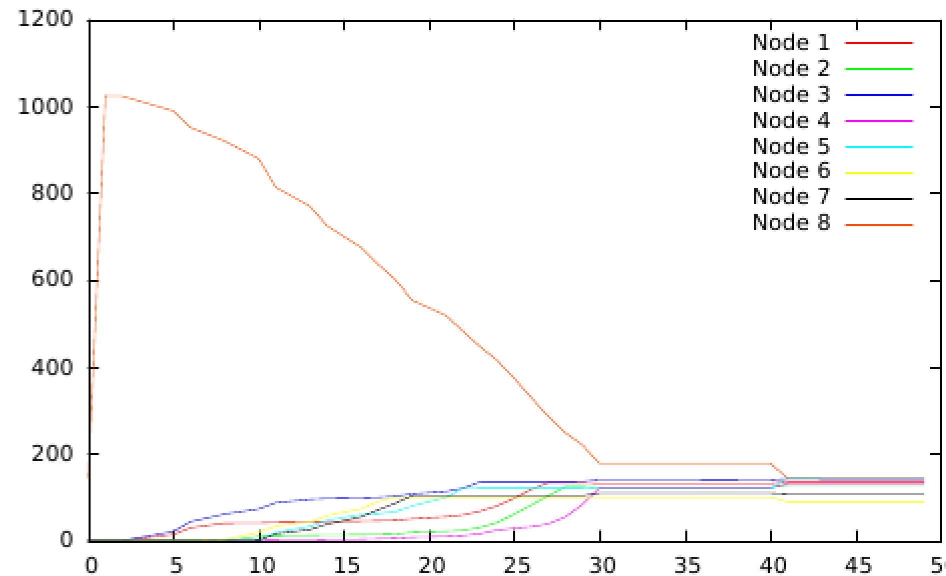
redhat.

AutoNUMA23 numa02_SMT









8 nodes