AutoNUMA

Red Hat, Inc.

Andrea Arcangeli
aarcange at redhat.com

1 Apr 2012

AutoNUMA components

- knuma_scand
  - If stopped, everything stops
  - Triggers the chain reaction when started
- NUMA hinting page faults
- knuma_migratedN (per node)
- scheduler (CPU follow memory & active idle balancing)
- Memory follow CPU (NUMA hinting page faults)
- False sharing detection (page->autonuma_last_nid)

AutoNUMA data

- sched_autonuma
  - task_struct (per-thread statistical NUMA info)
    - Generated by NUMA hinting page faults
- mm_autonuma
  - mm_struct (per-process statistical NUMA info)
    - Working set or ~RSS
    - Generated by knuma_scand

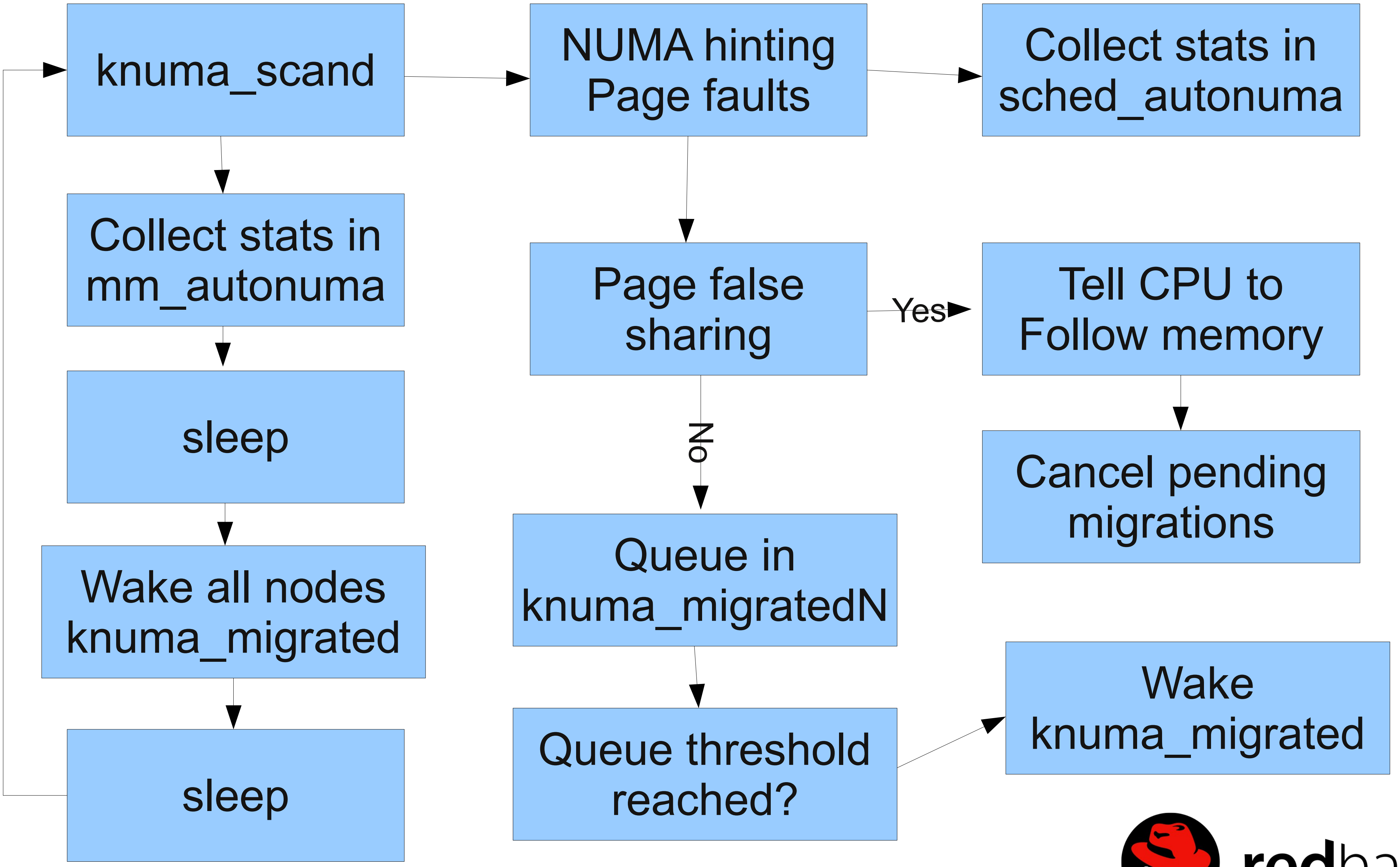
sched_autonuma

```
struct sched_autonuma {
        int autonuma_node;
        bool autonuma_stop_one_cpu;
        unsigned long numa_fault_pass;
        unsigned long numa_fault_tot;
        unsigned long numa_fault[0];
};
```

mm_autonuma

```
struct mm_autonuma {
        struct list_head mm_node;
        struct mm_struct *mm;
        unsigned long numa_fault_tot;
        unsigned long numa_fault_pass;
        unsigned long numa_fault[0];
};
```

AutoNUMA logic



knuma_scand

Collect stats in mm_autonuma

sleep

Wake all nodes knuma_migrated

sleep

NUMA hinting Page faults

Page false sharing

No

Queue in knuma_migratedN

Queue threshold reached?

Collect stats in sched_autonuma

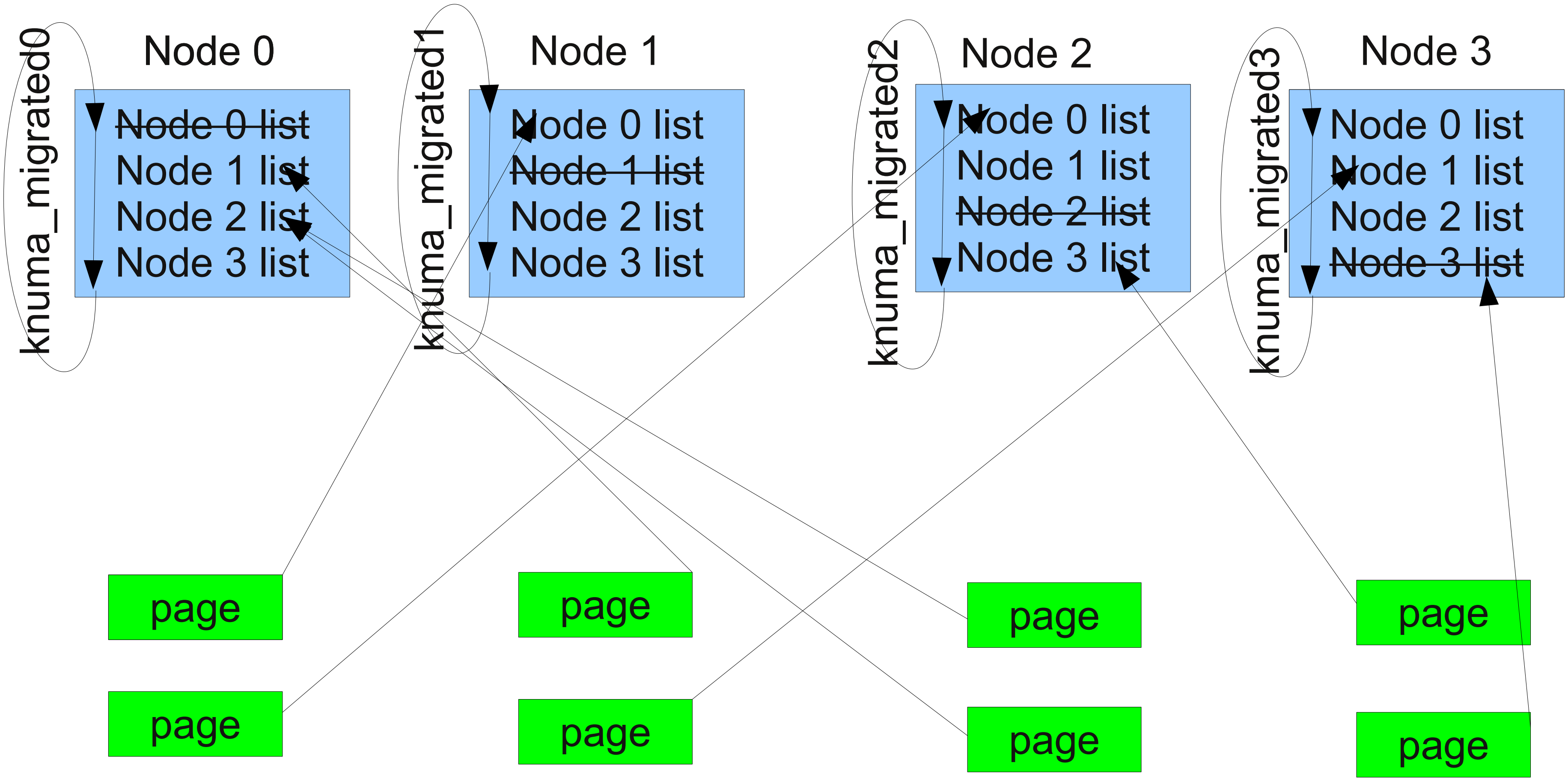
Yes

Tell CPU to Follow memory

Cancel pending migrations

Wake knuma_migrated

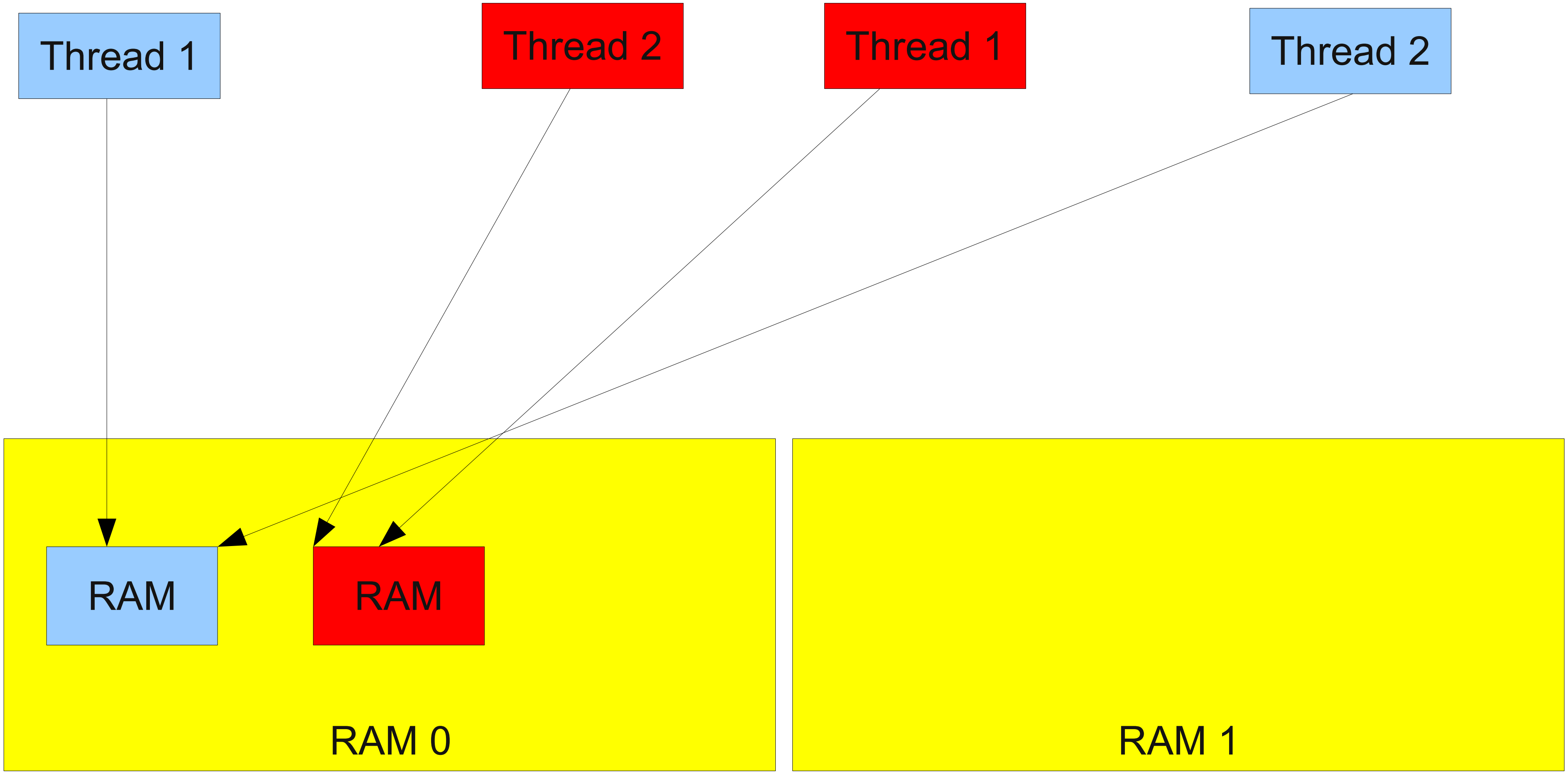
AutoNUMA knuma_migratedN



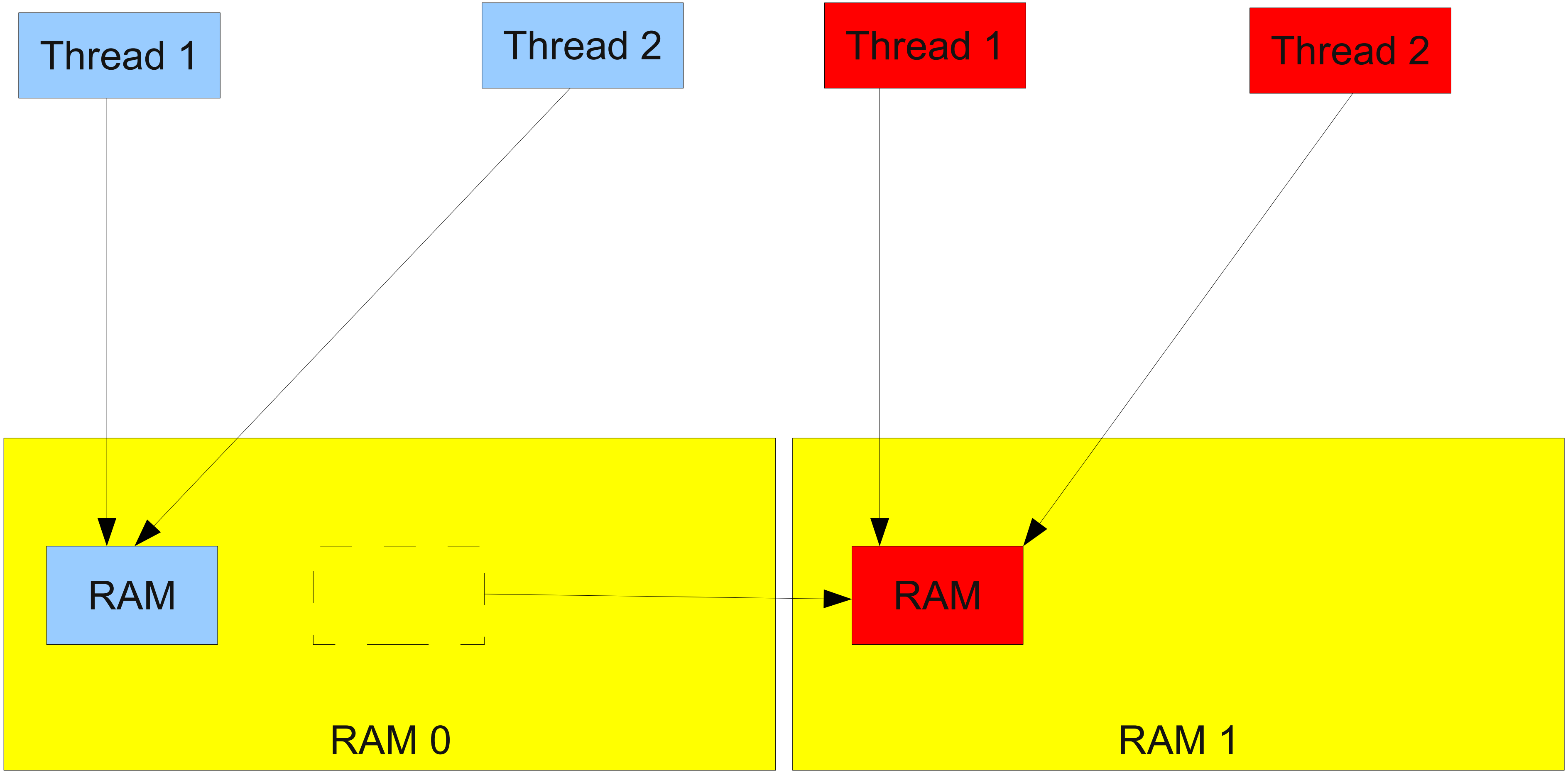
Hardware

- 2 NUMA nodes
- 2 CPU sockets
- 6 CPU cores per socket
- 2 HT CPU threads per core (total 24 CPUs)
- 8GB of RAM per node (total 16 GB of RAM)

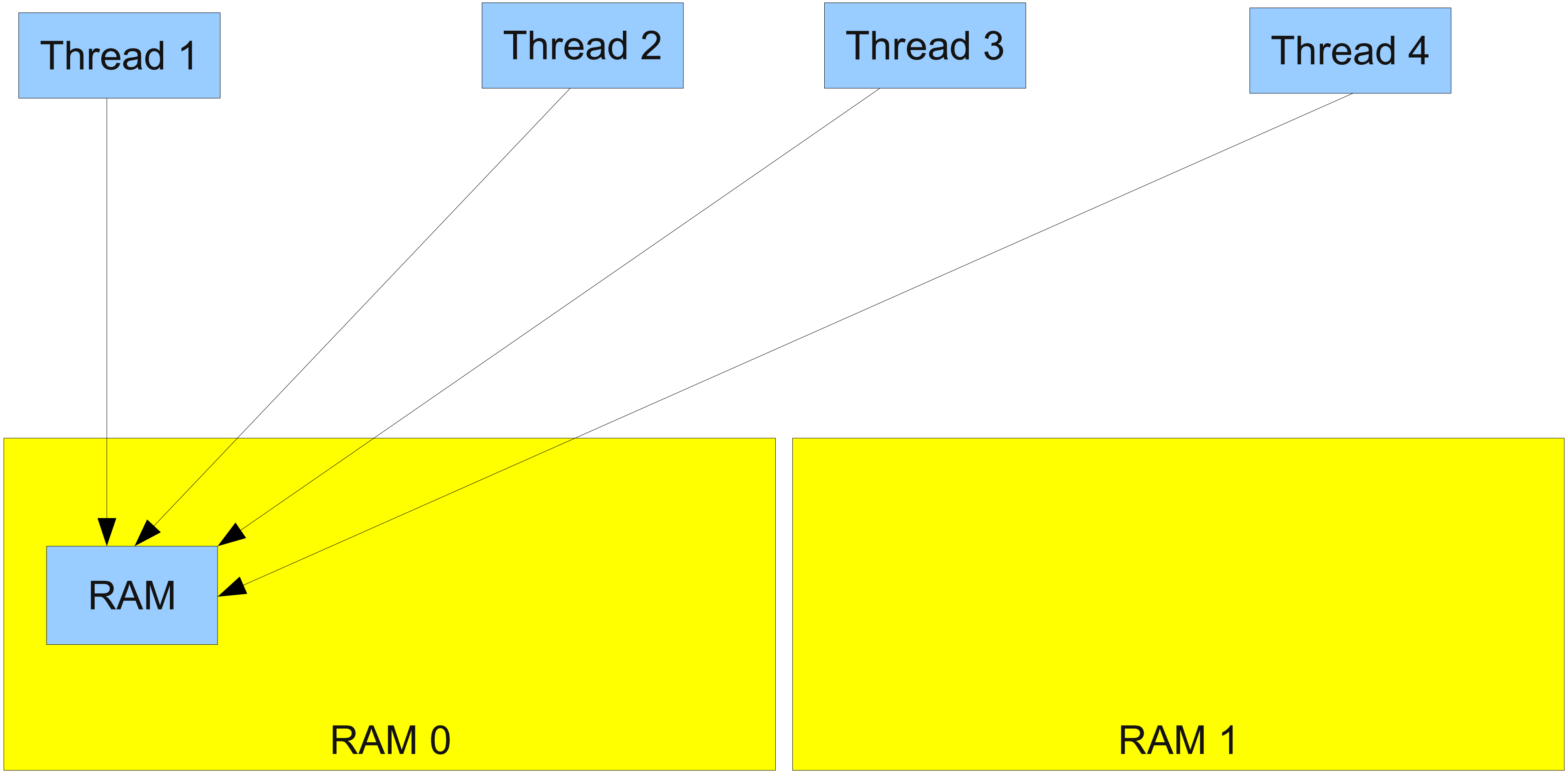
Numa01 (same node)



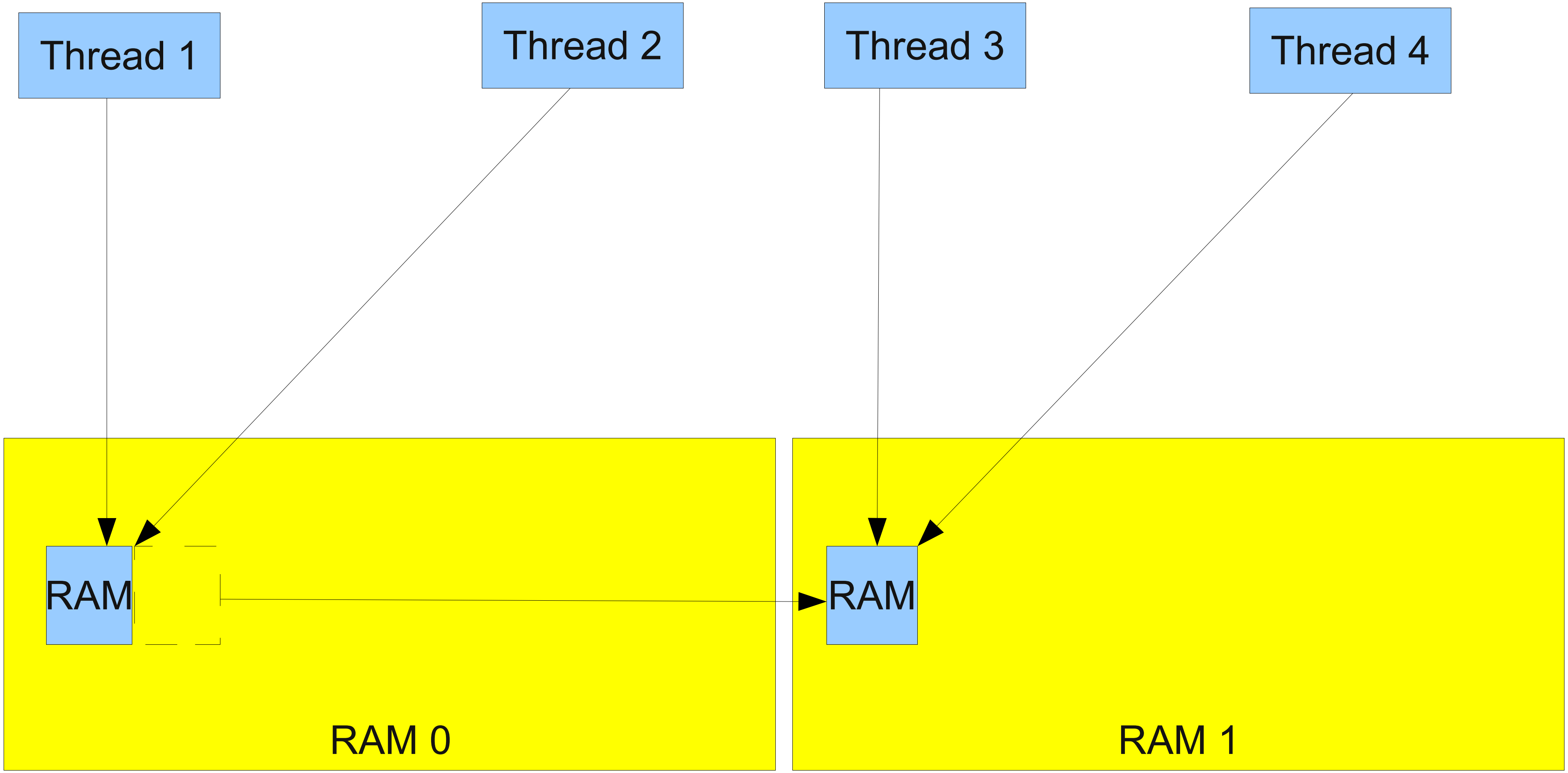
Numa01 (same node)



Numa02

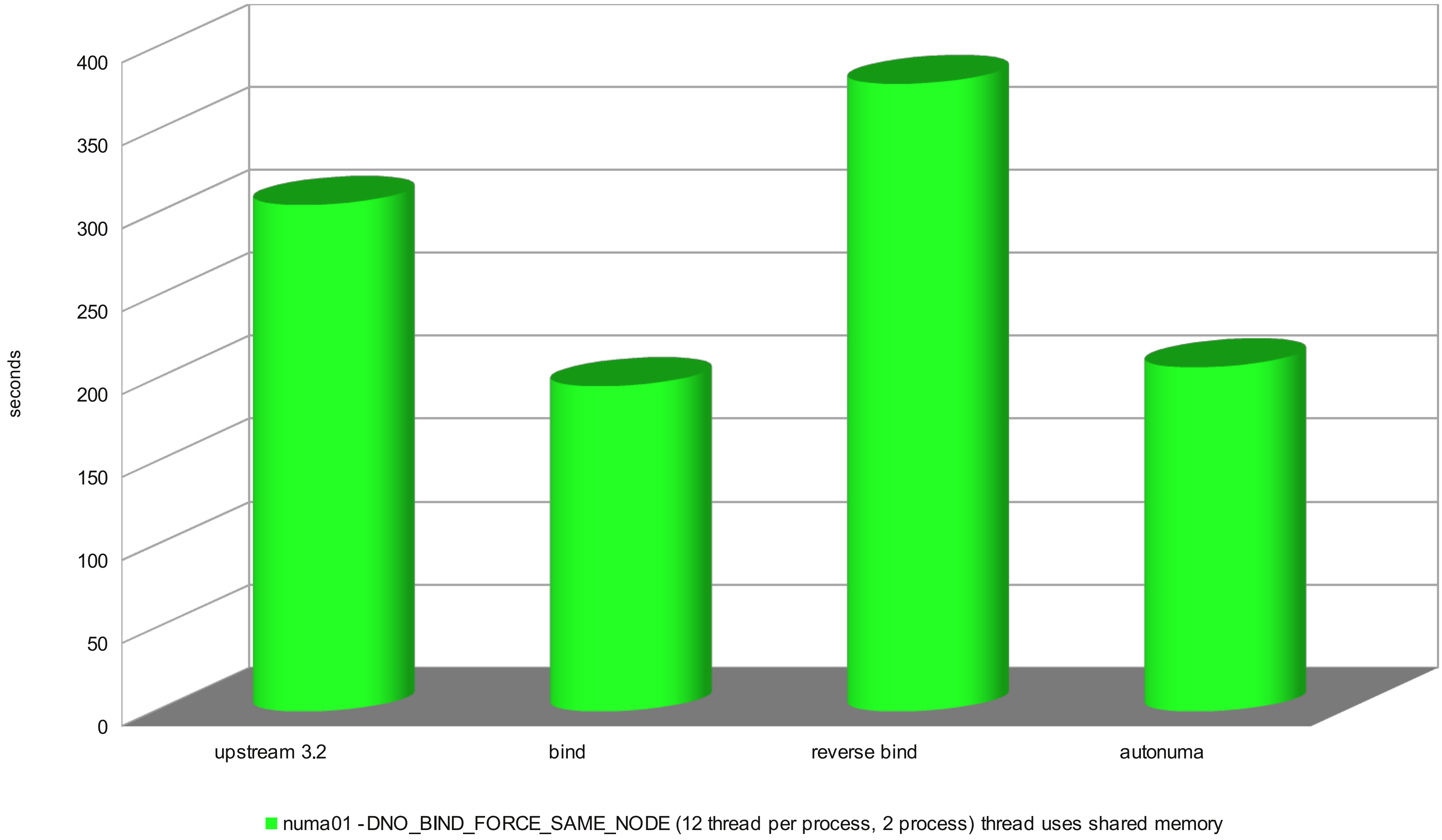


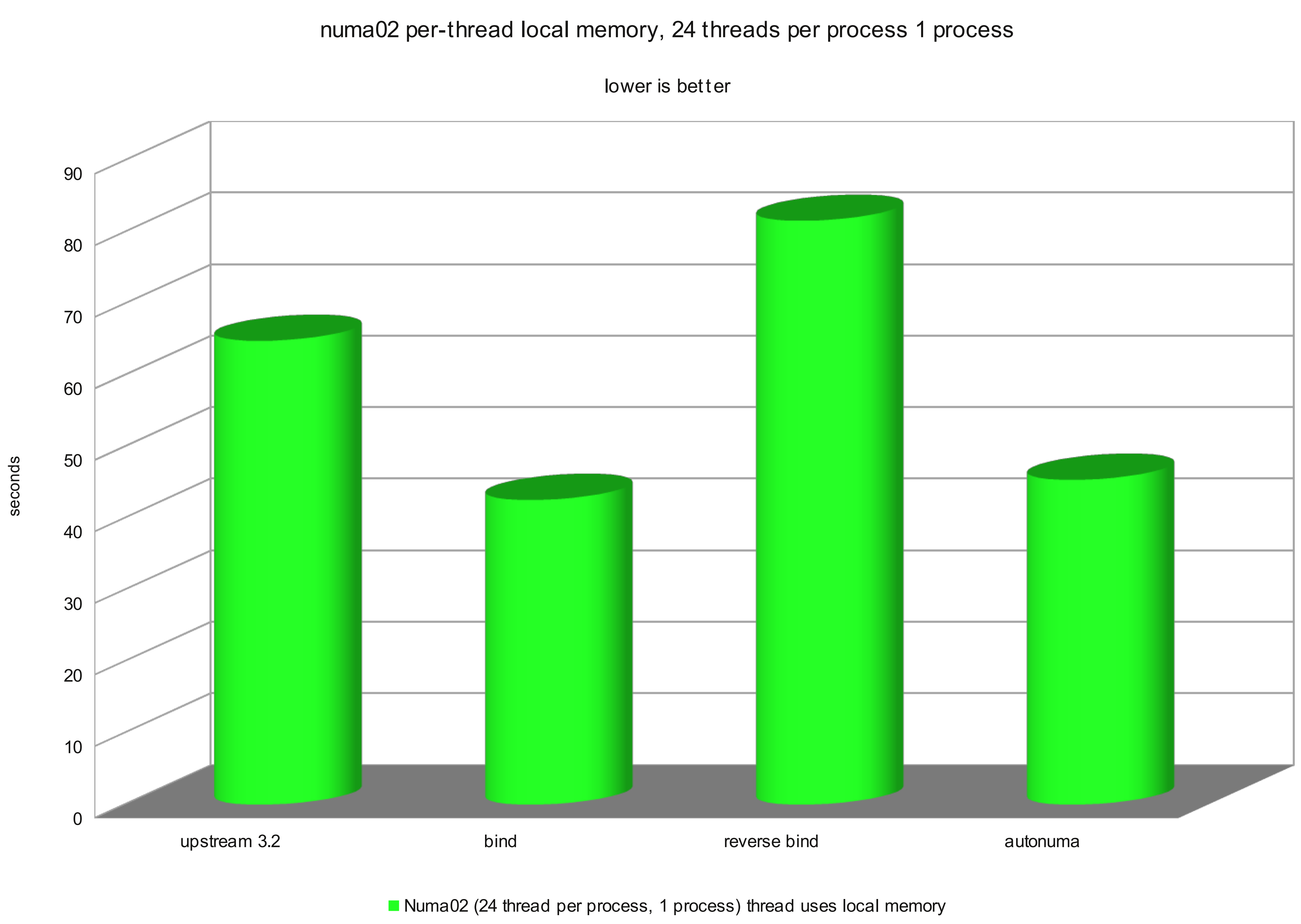
Numa02



numa01 -DNO_BIND_FORCE_SAME_NODE
all threads shares the same memory, 12 threads per process, 2 processes

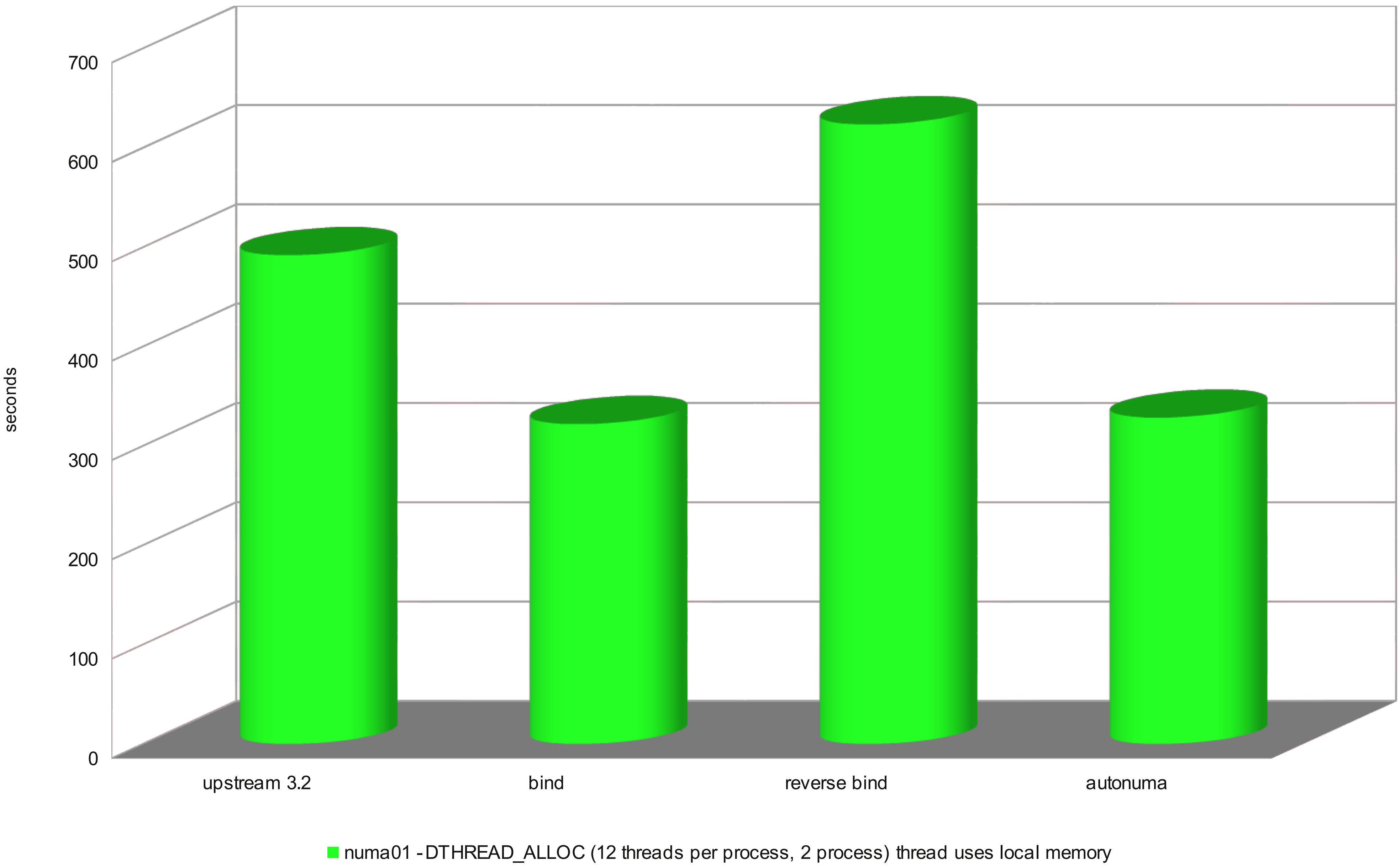
lower is better




numa02 per-thread local memory, 24 threads per process 1 process
lower is better
seconds
90
80
70
60
50
40
30
20
10
0
upstream 3.2
bind
reverse bind
autonuma
🟩 Numa02 (24 thread per process, 1 process) thread uses local memory

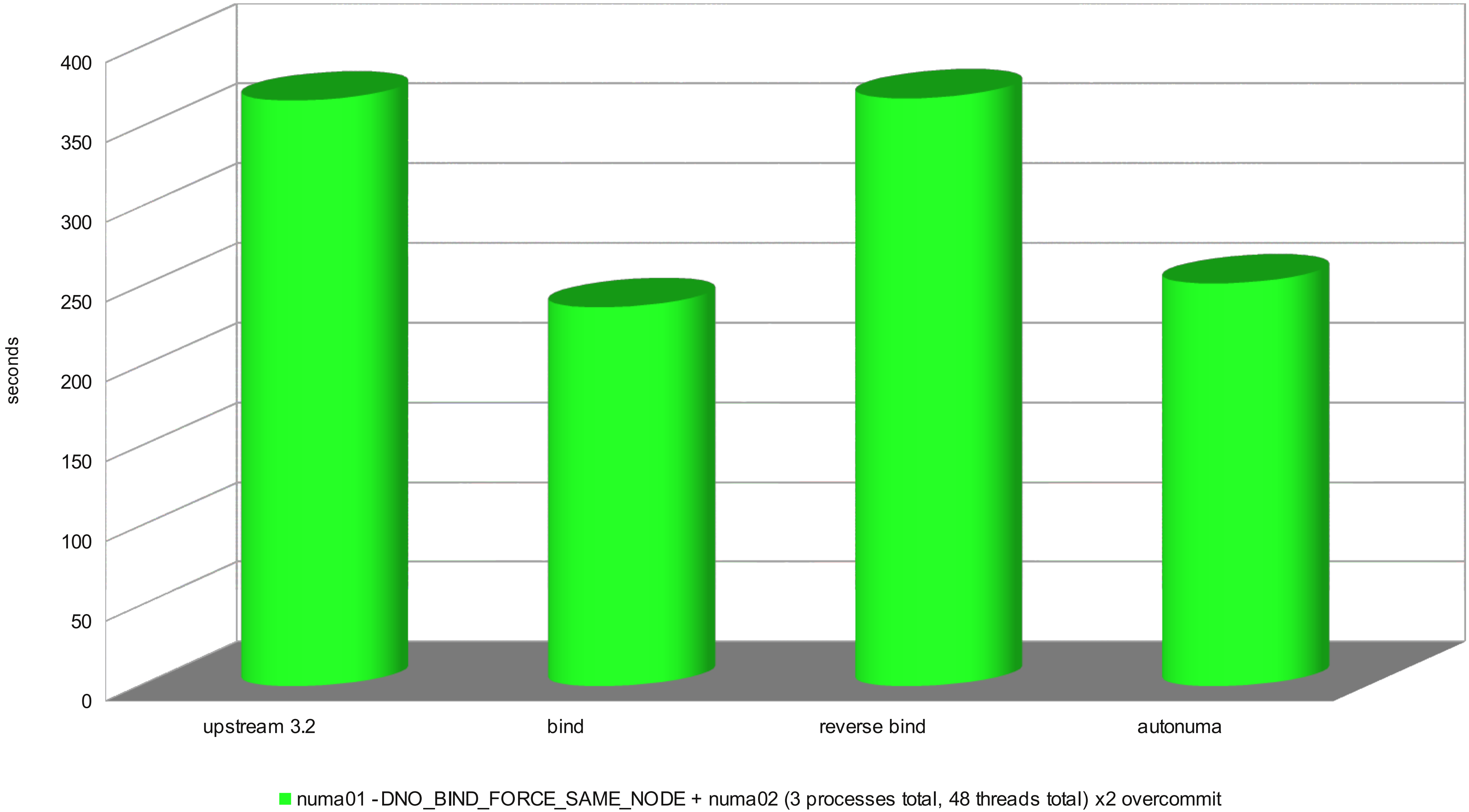
numa01 per-thread local memory, 12 threads per process, 2 processes

lower is better



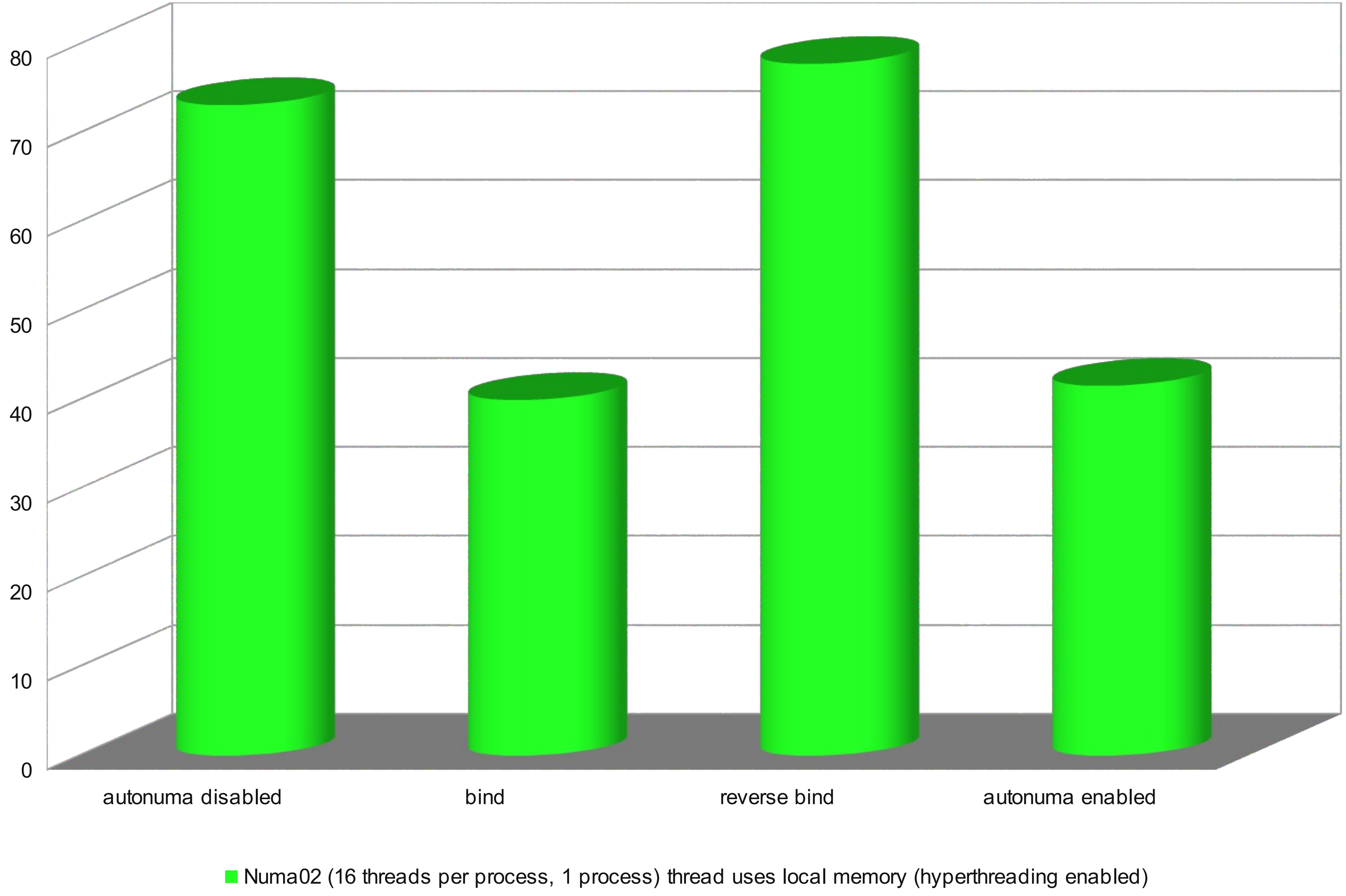
x2 CPU overcommit: numa01 -DNO_BIND_FORCE_SAME_NODE + numa02
24 threads using local memory +
12 threads using shared memory +
12 threads using shared memory

lower is better



numa02 per-thread local memory, 12 threads per process 1 process (HT enabled)
SMT testcase

lower is better



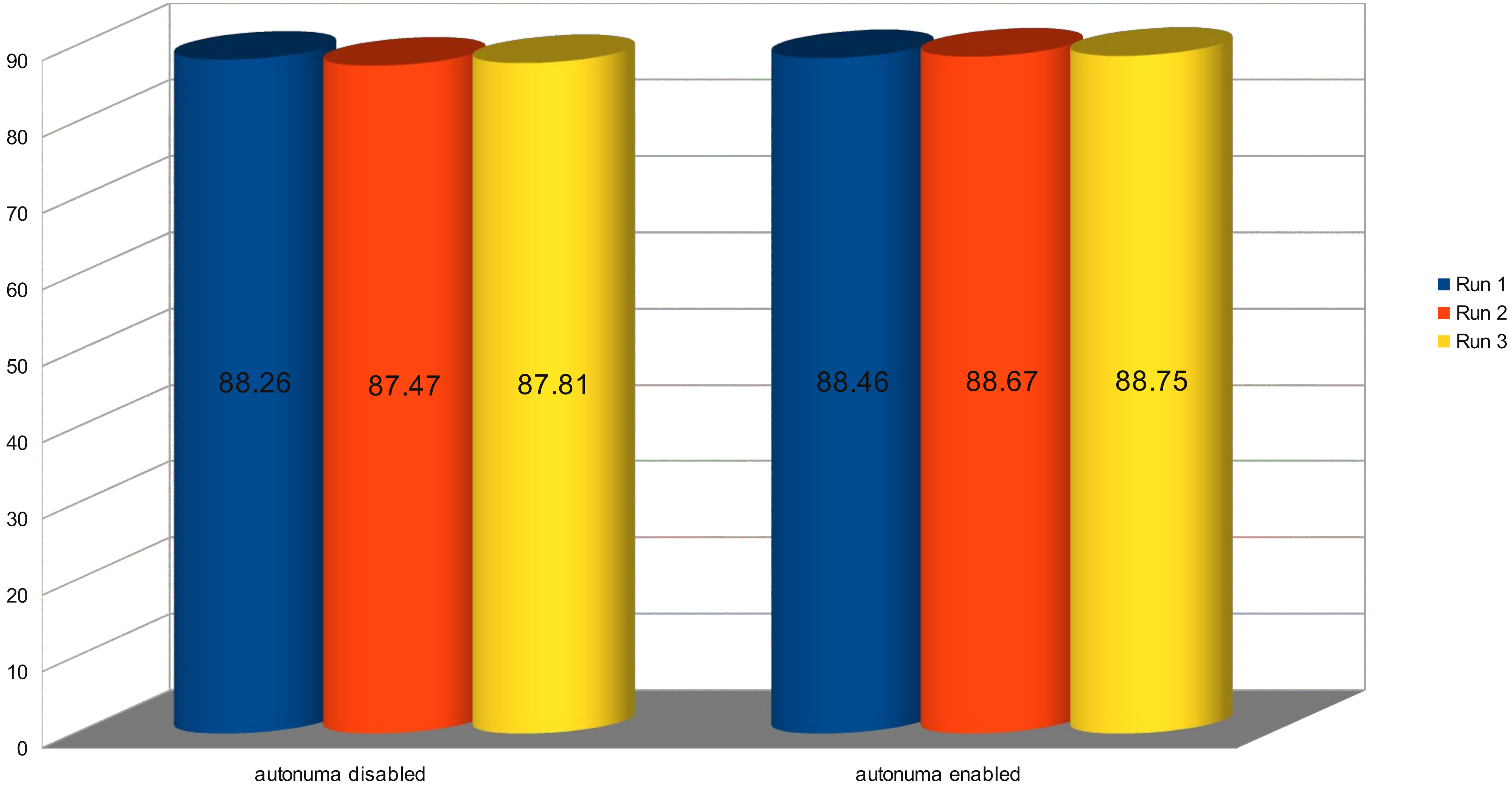
redhat

autonuma benchmark (hash 7e4dc3dbbda23b873ca7771b5cf296078e6ed1f7 vs 3.2 upstream default vs 3.2 upstream bind vs upstream inverse bind)				
	autonuma off	bind	reverse bind	autonuma
numa01 -DNO_BIND_FORCE_SAME_NODE (12 thread per process, 2 process) thread uses shared memory	305.36	196.07	378.34	207.47
Numa02 (24 thread per process, 1 process) thread uses local memory	64.81	42.58	81.6	45.39
numa01 -DTHREAD_ALLOC (12 threads per process, 2 process) thread uses local memory	491.88	321.94	623.62	328.43
numa01 -DNO_BIND_FORCE_SAME_NODE + numa02 (3 processes total, 48 threads total) x2 overcommit	366.96	237.43	368.35	252.31

Autonuma SMT fix uses hash 6e7267f0c9973f207a826c6b1fdae4e69c54ea80 Numa02 (16 threads per process, 1 process) thread uses local memory (hyperthreading enabled)	73.16	39.99	77.8	41.59

Kernel build time in seconds on tmpfs (make -j32)
Autonuma enabled includes one knuma_scand pass every 10sec

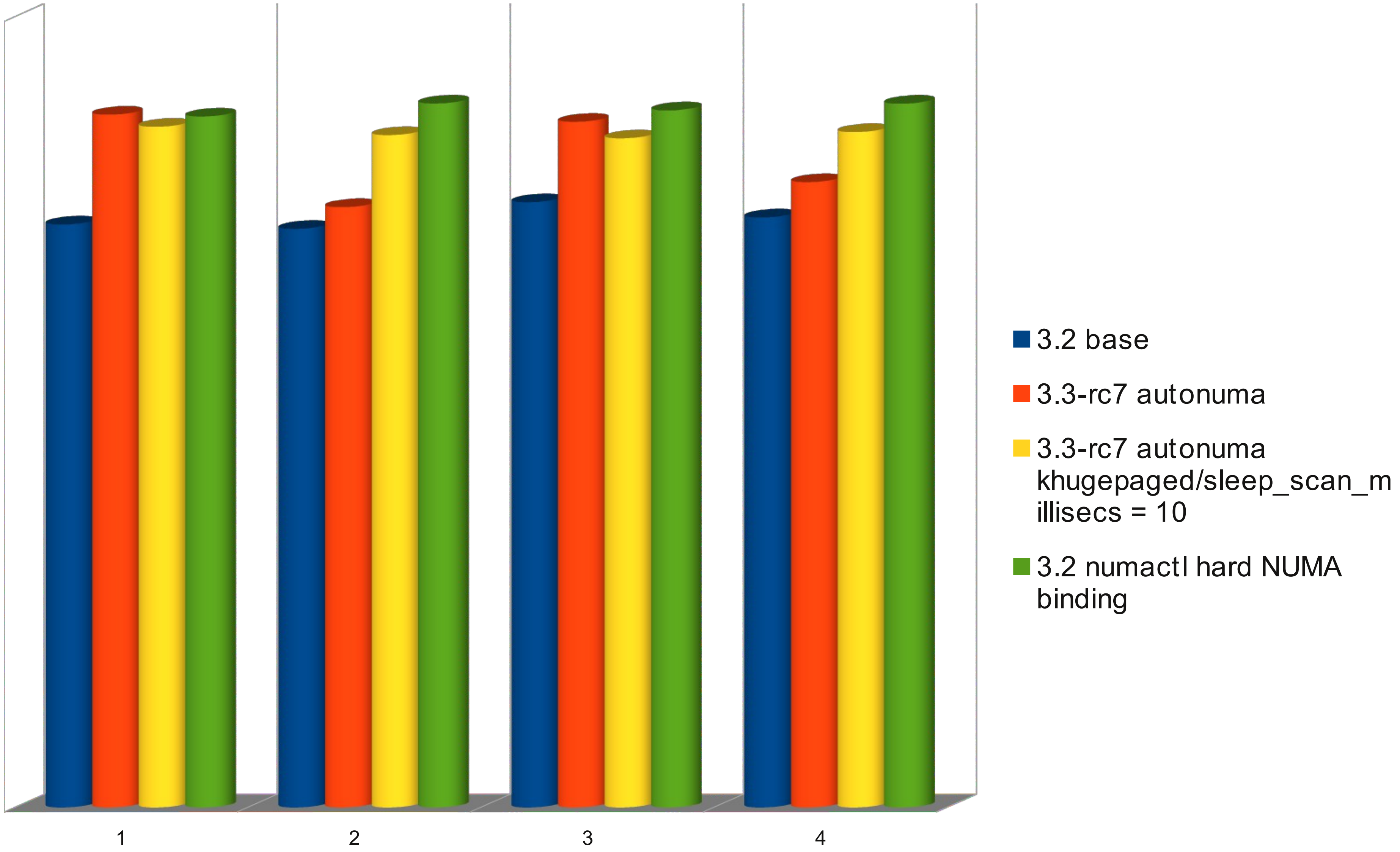
Worst possible case for AutoNUMA (gcc too short lived)
Average increase in build time 0.88%



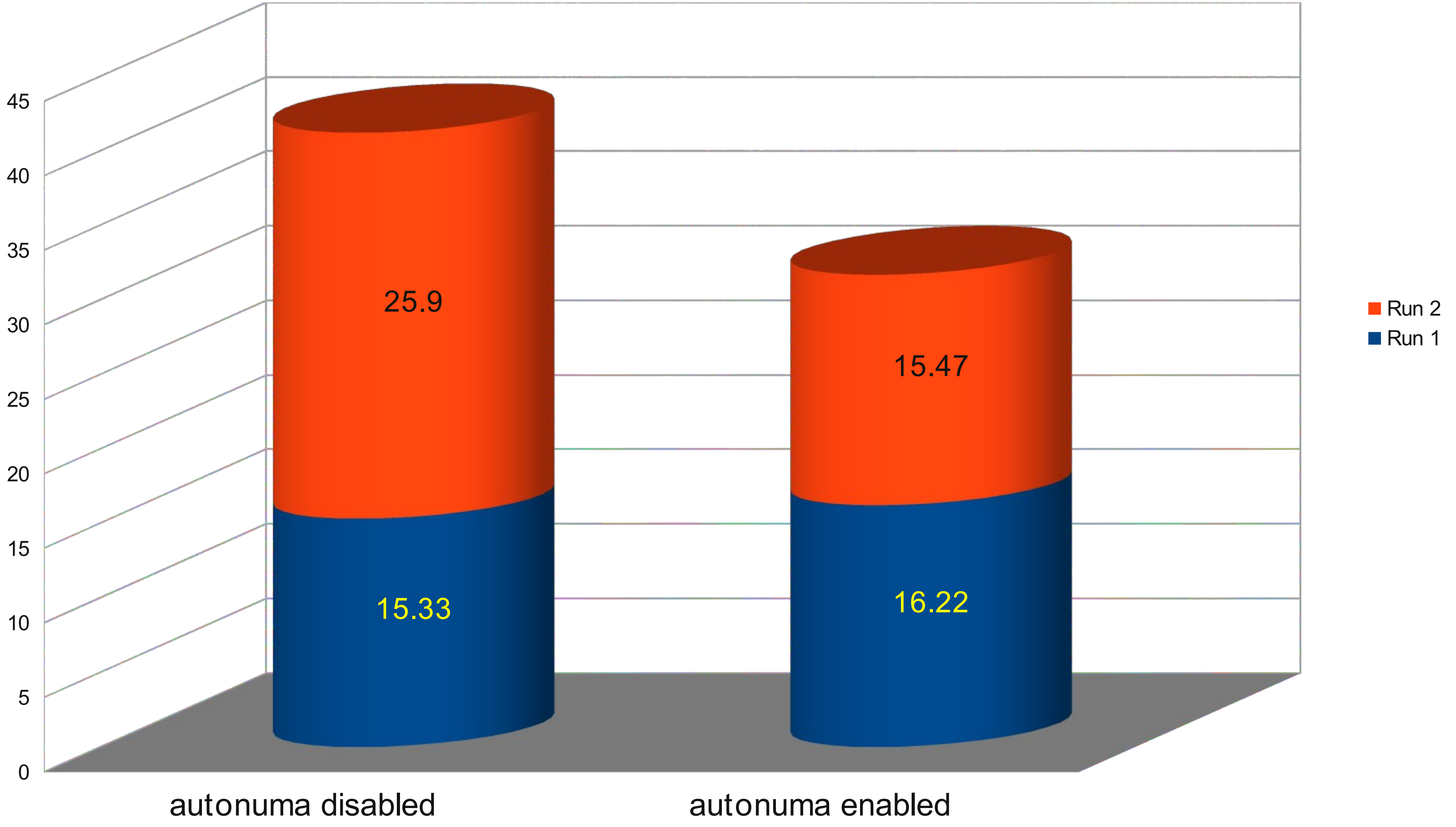
redhat

autonuma overhead kernel build tmpfs (make -j32)	Run 1	Run 2	Run 3
autonuma disabled	88.262	87.465	87.807
autonuma enabled	88.459	88.669	88.745

SPECjbb results 2 NUMA nodes, 8 CPUs per node, 16 CPUs total
THP enabled, no virt

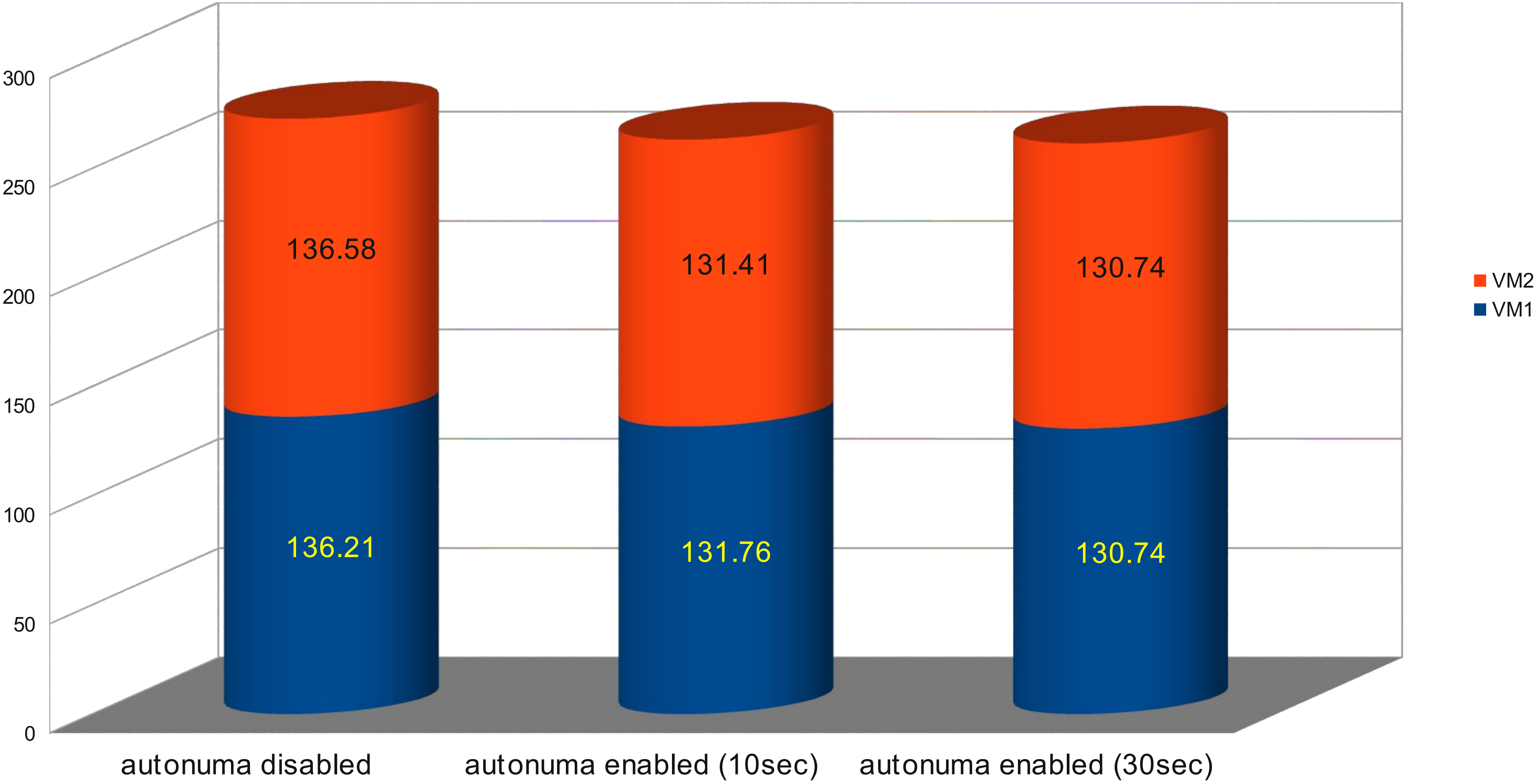


Virt guest "memhog -r100 1g" (autonuma includes 1 knuma_scand pass every 10 sec) KVM host autonuma enabled/disabled, THP enabled Guest VM fits in one host NUMA node



kernel build -j16 in parallel in 2 KVM (both in tmpfs, in a loop started in sync)
Both guest VM fits in one host NUMA node
autonuma/knuma_scand/scan_sleep_pass_millisecs = 5000 | 15000 (10sec | 30sec)

Host autonuma enabled/disabled, THP on, 12 vcpu per guest, 24 CPUs total on host



TODO: THP native migration

- THP native migration
  - SPECjbb results with khugepaged boosted shows the main bottleneck left is lack of THP native migration:
    - One copy in migration
    - One copy in khugepaged to rebuild the hugepage
  - Once this feature is added, AutoNUMA should perform even closer to numactl than it does now with khugepaged boosted (3rd column for every SPECjbb pass).
  - Urgent

TODO: scheduler

- Reduce autonuma_balance invocation frequency
  - Possibly run it from softirq like the load balance
- Possibly integrate it more closely into CFS

TODO: struct page

- Allocate the 24 bytes per page only when booted on NUMA hardware

TODO: document sched/numa.c

- And write proper high level documentation to put in Documentation/vm/autonuma.txt .