

AutoNUMA bench

Red Hat, Inc.

Andrea Arcangeli
aarcange at redhat.com

```
git clone --reference linux -b autonuma git://git.kernel.org/pub/scm/linux/kernel/git/andrea/aa.git  
c9effed1071202090cb9e09da814dabf1b9af409
```

21 Mar 2012



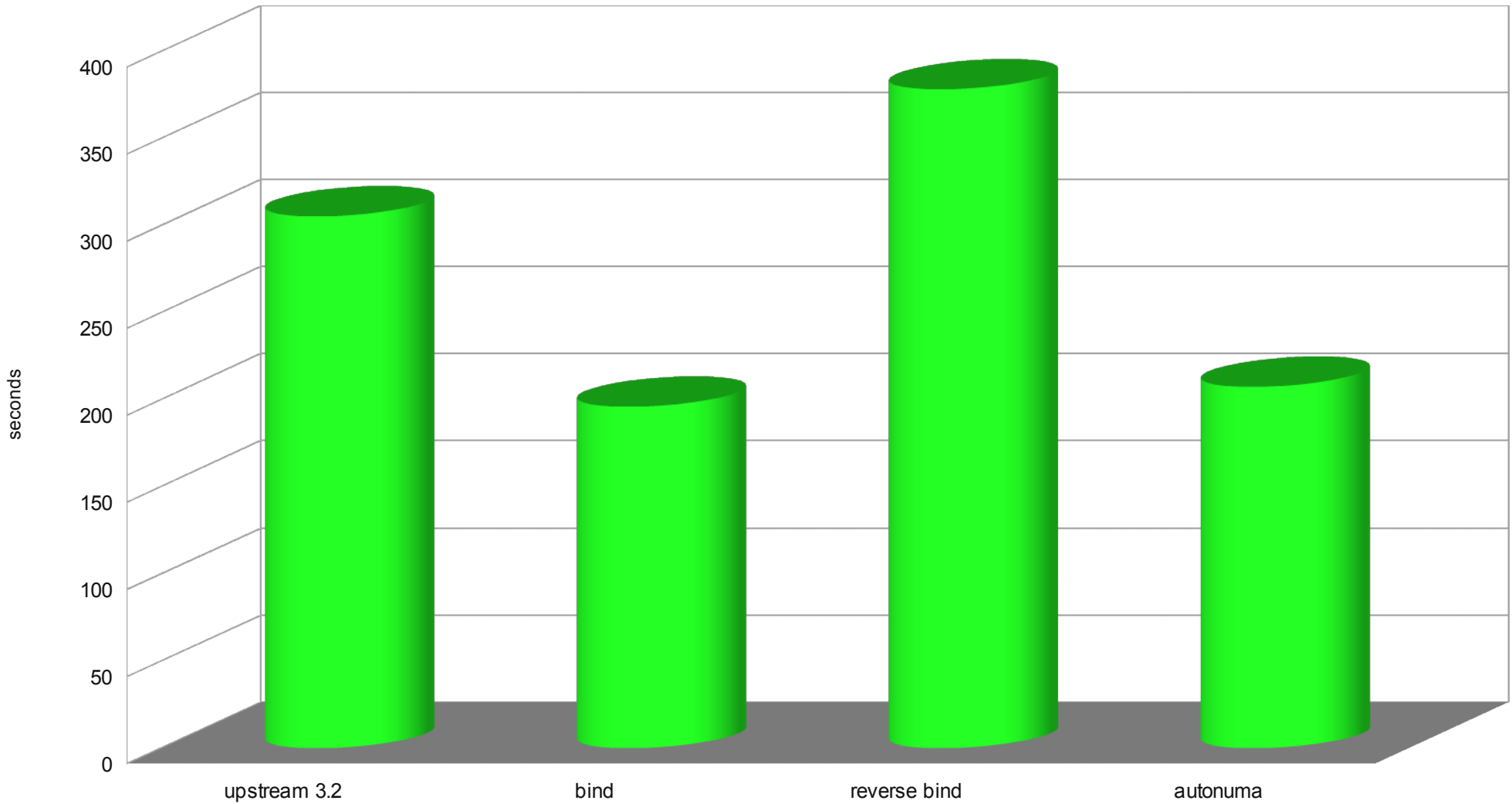
Hardware

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

lower is better

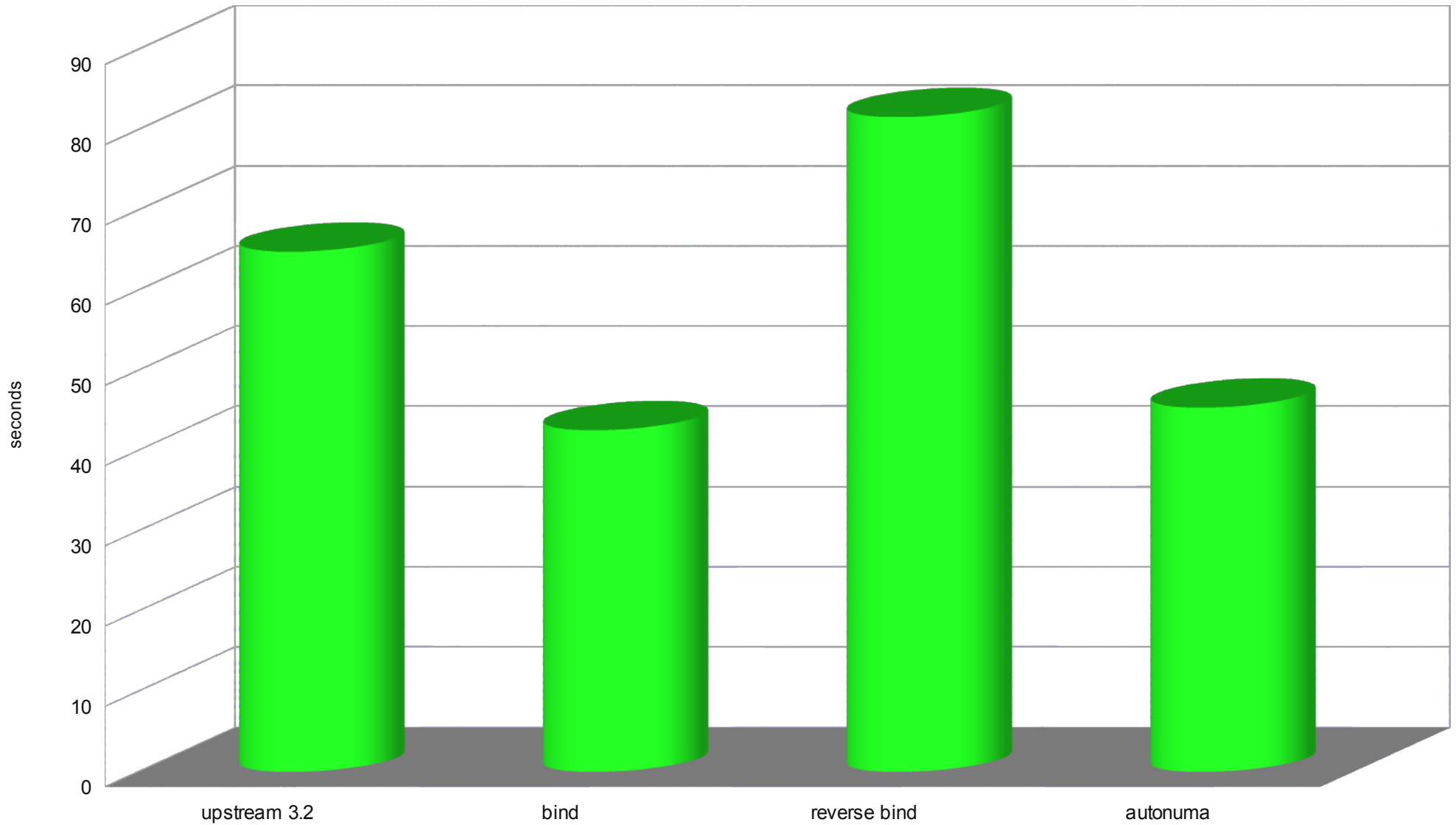


■ numa01 -DNO_BIND_FORCE_SAME_NODE (12 thread per process, 2 process) thread uses shared memory



numa02 per-thread local memory, 24 threads per process 1 process

lower is better

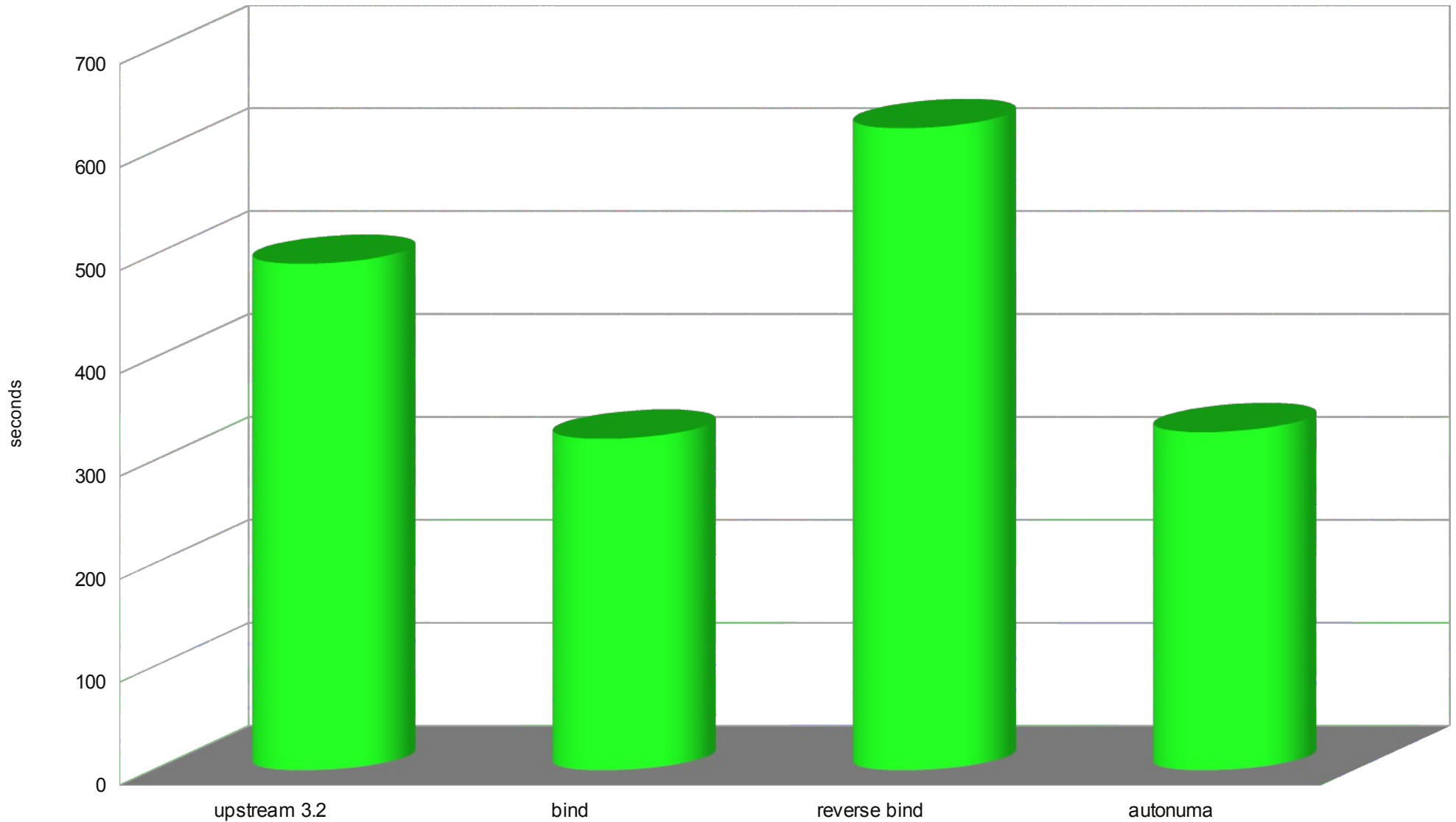


■ Numa02 (24 thread per process, 1 process) thread uses local memory



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

lower is better

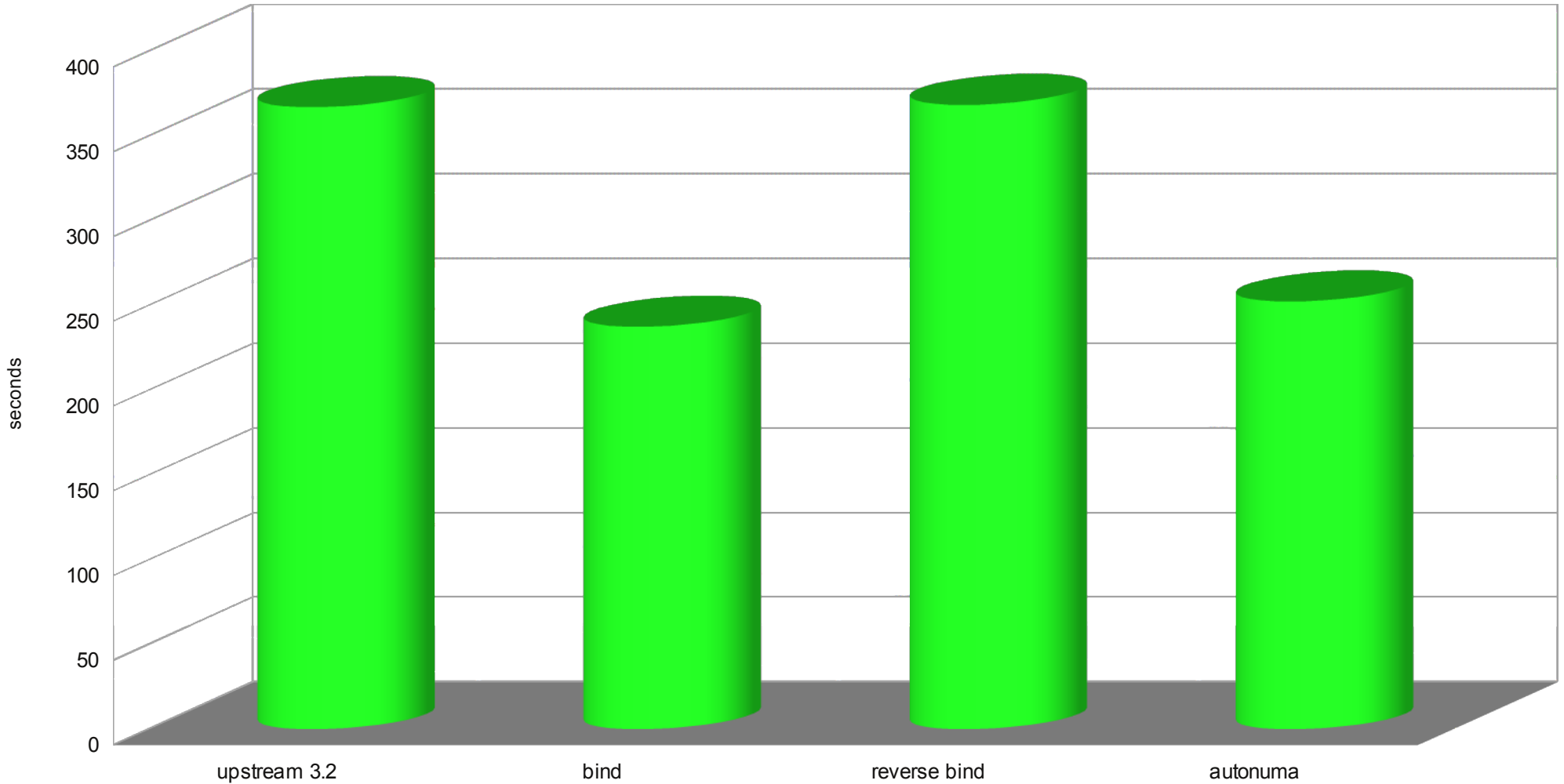


■ numa01 -DTHREAD_ALLOC (12 threads per process, 2 process) thread uses local memory



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



■ numa01 -DNO_BIND_FORCE_SAME_NODE + numa02 (3 processes total, 48 threads total) x2 overcommit

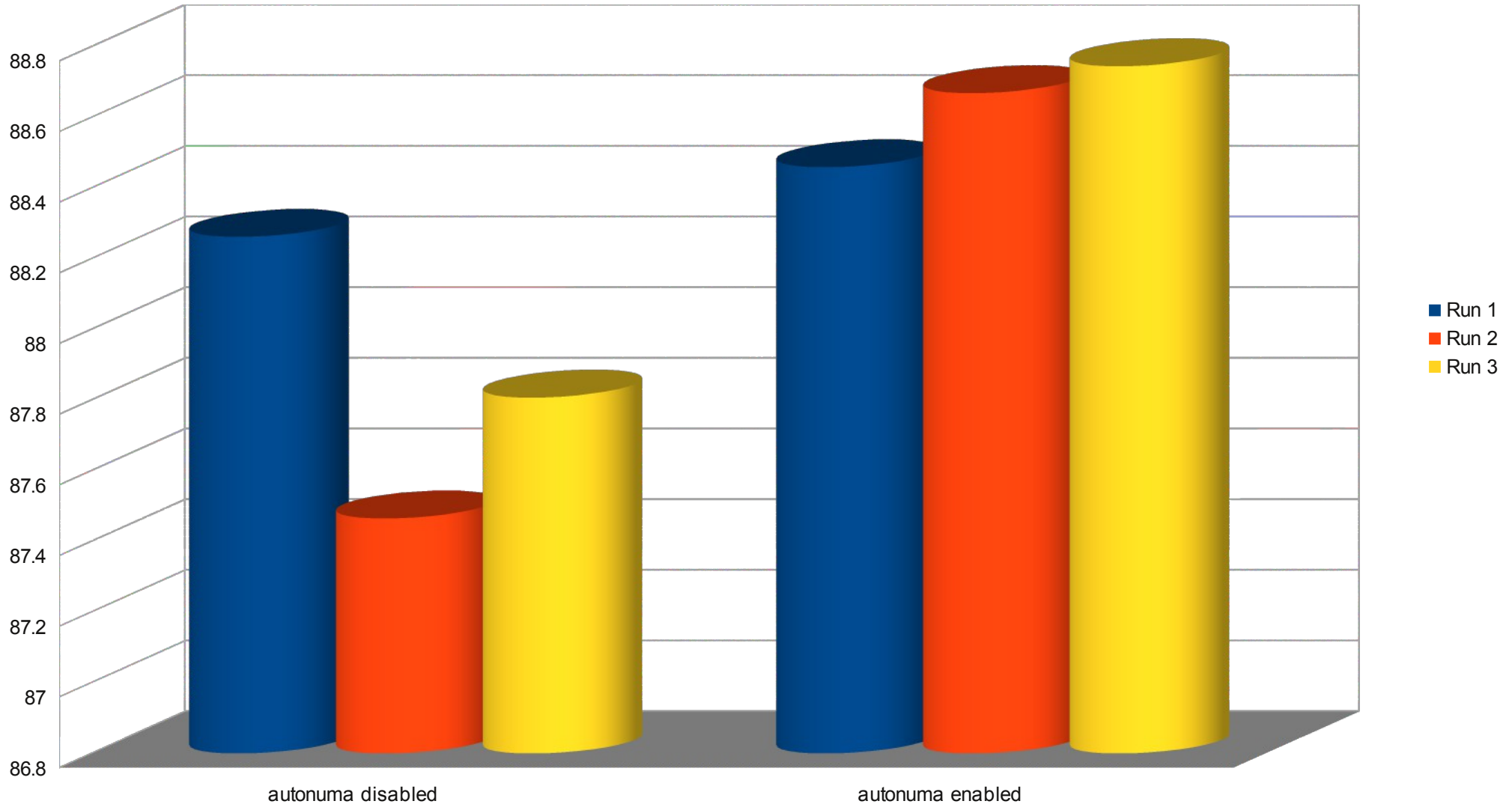


autonuma benchmark (hash 7e4dc3dbbda23b873ca7771b5cf296078e6ed1f7 vs 3.2 upstream default vs 3.2 upstream bind vs upstream inverse bind)	upstream 3.2	bind	rever se bind	auton uma
numa01 -DNO_BIND_FORCE_SAME_NODE (12 thread per process, 2 process) thread uses shared memory	305.36	196.0 7	378.3 4	207.4 7
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	321.9 4	623.6 2	328.4 3
numa01 -DNO_BIND_FORCE_SAME_NODE + numa02 (3 processes total, 48 threads total) x2 overcommit	366.96	237.4 3	368.3 5	252.3 1



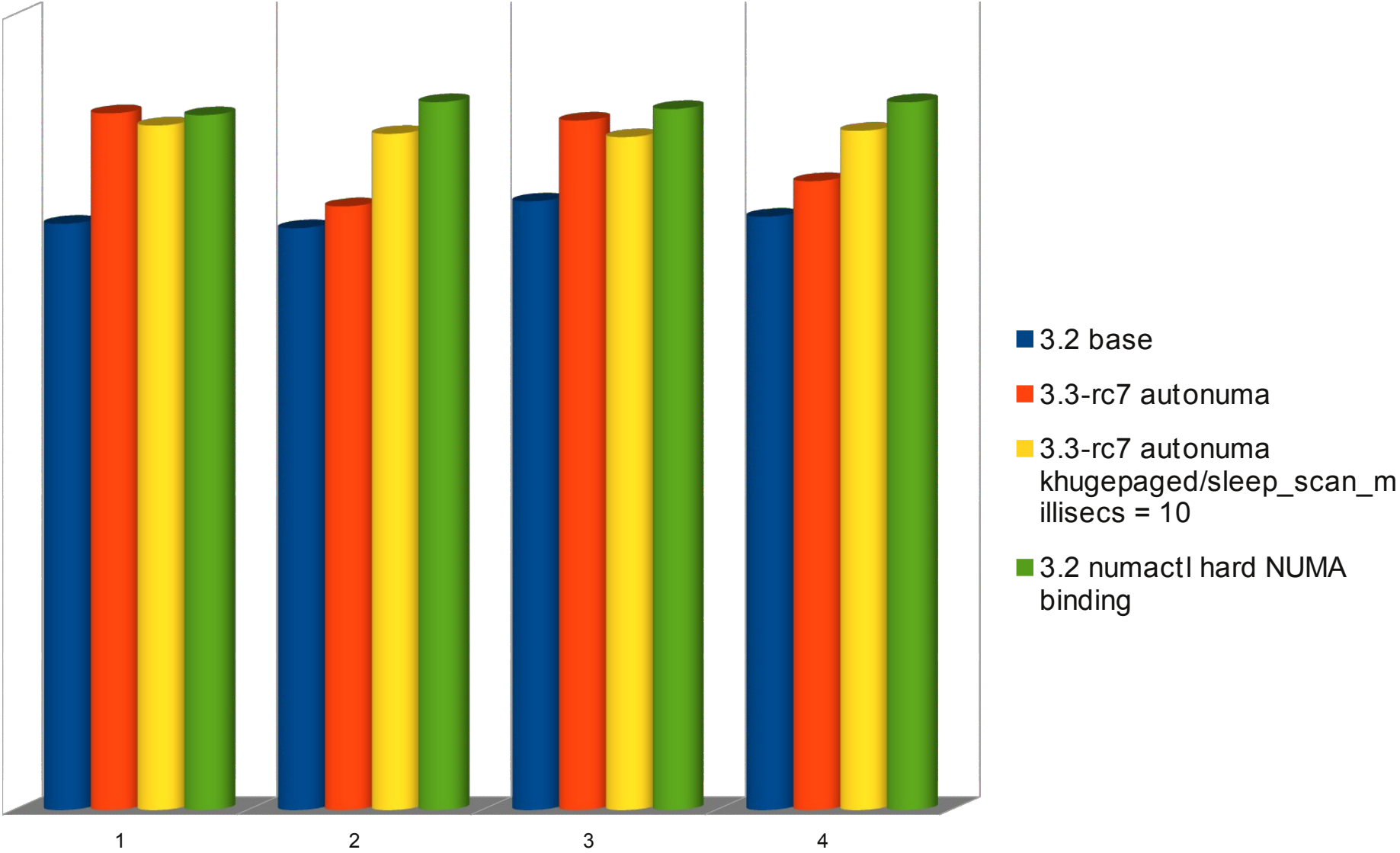
Kernel build time in seconds on tmpfs (make -j32)

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



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



redhat.

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: struct page

- Allocate the 24 bytes per page only when booted on NUMA hardware like memcg does it:
 - Tricky with the different direct mapping implementation
 - At build time struct page is already not enlarged if `CONFIG_AUTONUMA=n`
 - Not too urgent
- No need to shrink the 24 bytes. Perhaps we could save 8 bytes, by crippling the list implementation (like forbidding list-deletion operations), but list-deletion (for migration cancellation when false sharing is detected) is already used.



TODO: document sched/numa.c

- Asked by Hillf as urgent item.
- Less urgent: write proper high level documentation to put in Documentation/vm/autonuma.txt .



TODO: split patches

- Split in smaller patches.