AutoNUMA

Red Hat, Inc.

Andrea Arcangeli
aarcange at redhat.com

30 Aug 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
AutoNUMA data

➢ sched_autonuma
   ➢ task_struct

➢ mm_autonuma
   ➢ mm_struct
      ➢ Working set or ~RSS
AutoNUMA logic

1. `knuma_scand`
2. Collect stats in `mm_autonuma`
3. `sleep`
4. Wake all nodes `knuma_migrated`
5. `sleep`
6. `NUMA hinting Page faults`
7. `Page false sharing`
8. Queue in `knuma_migrated`
9. Queue threshold reached?
10. Yes
    - Tell CPU to Follow memory
    - Cancel pending migrations
    - Wake `knuma_migrated`
11. No
    - Collect stats in `sched_autonuma`
AutoNUMA knuma_migratedN

Node 0
- Node 0 list
- Node 1 list
- Node 2 list
- Node 3 list

Node 1
- Node 0 list
- Node 1 list
- Node 2 list
- Node 3 list

Node 2
- Node 0 list
- Node 1 list
- Node 2 list
- Node 3 list

Node 3
- Node 0 list
- Node 1 list
- Node 2 list
- Node 3 list

page
page
page
page
page
page
page
page

Copyright © 2012 Red Hat Inc.
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)
numa01 (converged)
numa02 (startup)
numa02 (converged)
autonuma-benchmark

$ git clone git://gitorious.org/autonuma-benchmark/autonuma-benchmark.git
$ cd autonuma-benchmark
$ sudo ./start_bench.sh -s -t
4 runs, average and stdev

Lower is better

2 nodes

Copyright © 2012 Red Hat Inc.
4 runs of each test, all results

Lower is better

2 nodes

Copyright © 2012 Red Hat Inc.
4 runs, average and stdev

Lower is better

8 nodes

Seconds

numa01 | numa01_THREAD_ALLOC | numa02 | numa02_SMT
--- | --- | --- | ---
AutoNUMA23 | 1000 | 1000 | 1000
3.6-rc1 | 4000 | 4000 | 4000
4 runs of each test, all results

Lower is better

Seconds

 numa01  numa01_THREAD_ALLOC  numa02  numa02_SMT

AutoNUMA23 run1  AutoNUMA23 run2  AutoNUMA23 run3  AutoNUMA23 run4

3.6-rc1 run1  3.6-rc1 run2  3.6-rc1 run3  3.6-rc1 run4

8 nodes
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 numa02_SMT

2 nodes
sched-numa-rewrite numa02

2 nodes
3.6-rc1 numa01_THREAD_ALLOC

2 nodes
2 nodes
AutoNUMA23 numa01

8 nodes → heavy false sharing
AutoNUMA23 numa02

Copyright © 2012 Red Hat Inc.

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

Worst possible case for AutoNUMA (gcc too short lived)
Average increase in build time 0.88%
SPECjbb results 2 NUMA nodes, 8 CPUs per node, 16 CPUs total
THP enabled, no virt
Higher is better

- 3.2 base
- 3.3-rc7 autonuma
- 3.3-rc7 autonuma
  khugepaged/sleep_scan_millisecs = 10
- 3.2 numactl hard NUMA binding
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  
Lower is better
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)
Lower is better

Host autonuma enabled/disabled, THP on, 12 vcpu per guest, 24 CPUs total on host
KVM/KSM/THP bench run by IBM

SPECjbb2005 on Linux 3.4-rc2

1. Higher is better
2. Can’t compare VM1 scores with VM2/VM3 scores.
3. Can compare VM2 score with VM3 score (should be equal)
4. One Node = 12 GB (slightly > 12 GB) with 6 cores + 6 hyperthreads

- VM 1 (~ 1 node: 12 GB/14 GB with 12 vCPUs)
- VM 2 (4 GB/3 GB with 6 vCPUs)
- VM 3 (4 GB/3 GB with 6 vCPUs)