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
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];
};
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

1. **knuma_scand**
   - Collect stats in `mm_autonuma`
   - Sleep
   - Wake all nodes `knuma_migrated`
   - Sleep

2. **NUMA hinting Page faults**
   - Page false sharing
   - Queue in `knuma_migrated`
   - Queue threshold reached?
     - Yes: Tell CPU to Follow memory
     - No: Cancel pending migrations
   - Wake `knuma_migrated`

3. **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

page
page
page
page

Copyright © 2012 Red Hat Inc.
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)

Thread 1

Thread 2

Thread 1

Thread 2

RAM 0

RAM 1

RAM
Numa02

Thread 1

Thread 2

Thread 3

Thread 4

RAM

RAM 0

RAM 1
Numa02

Thread 1

Thread 2

Thread 3

Thread 4

RAM 0

RAM 1

Copyright © 2011 Red Hat Inc.
numa01 -DNO_BIND_FORCESAME_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

- 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

upstream 3.2 | bind | reverse bind | autonuma

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

[Bar chart showing performance metrics for different scenarios: upstream 3.2, bind, reverse bind, autonuma. Each bar represents the time taken in seconds, with lower values indicating better performance.]
numa02 per-thread local memory, 12 threads per process 1 process (HT enabled)

SMT testcase

lower is better

Numa02 (16 threads per process, 1 process) thread uses local memory (hyperthreading enabled)
<table>
<thead>
<tr>
<th>Configuration</th>
<th>autonuma off</th>
<th>bind</th>
<th>reverse bind</th>
<th>autonuma</th>
</tr>
</thead>
<tbody>
<tr>
<td>numa01 -DNO_BIND_FORCE_SAME_NODE (12 thread per process, 2 process) thread uses shared memory</td>
<td>305.36</td>
<td>196.0</td>
<td>378.34</td>
<td>207.47</td>
</tr>
<tr>
<td>Numa02 (24 thread per process, 1 process) thread uses local memory</td>
<td>64.81</td>
<td>42.58</td>
<td>81.6</td>
<td>45.39</td>
</tr>
<tr>
<td>numa01 -DTHREAD_ALLOC (12 threads per process, 2 process) thread uses local memory</td>
<td>491.88</td>
<td>321.9</td>
<td>623.62</td>
<td>328.43</td>
</tr>
<tr>
<td>numa01 -DNO_BIND_FORCE_SAME_NODE + numa02 (3 processes total, 48 threads total) x2 overcommit</td>
<td>366.96</td>
<td>237.4</td>
<td>368.35</td>
<td>252.31</td>
</tr>
</tbody>
</table>

| Configuration                                                                 | | | | |
| 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 k numa_scand pass every 10sec

Worst possible case for AutoNUMA (gcc too short lived)
Average increase in build time 0.88%
<table>
<thead>
<tr>
<th>autonuma overhead kernel build tmpfs (make -j32)</th>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>autonuma disabled</td>
<td>88.262</td>
<td>87.465</td>
<td>87.807</td>
</tr>
<tr>
<td>autonuma enabled</td>
<td>88.459</td>
<td>88.669</td>
<td>88.745</td>
</tr>
</tbody>
</table>
SPECjbb results 2 NUMA nodes, 8 CPUs per node, 16 CPUs total
THP enabled, no virt

- 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

Run 1
Run 2

25.9
15.33

15.47
16.22

autonuma disabled
autonuma enabled
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.