Reaching "EPYC" Virtualization Performance
Case Study: Tuning VMs for Best Performance on AMD EPYC 7002 / 7742 Processor Series Based Servers

Dario Faggioli <dfaggioli@suse.com>
Software Engineer - Virtualization Specialist, SUSE
GPG: 4B9B 2C3A 3DD5 86BD 163E 738B 1642 7889 A5B8 73EE
https://about.me/dario.faggioli
https://www.linkedin.com/in/dfaggioli/
https://twitter.com/DarioFaggioli (@DarioFaggioli)
A.K.A.: Pinning the vCPUs is enough, right?
AMD EPYC 7002 Series ("EPYC2")

AMD64 SMP SoC, EPYC family
Multi-Chip Module, 9 dies:
• 1 I/O die, off-chip communications (memory, other sockets, I/O)
• 8 "compute" dies (CCDs)
  – Core CompleX (CCX) → 4 cores (8 threads), its own L1-L3 cache hierarchy
  – Core Complex Die (CCD) == 2 CCXs: 8 cores (16 threads) + dedicated Infinity Fabric link to IO die

64 cores (128 threads), 2 socket, 8 memory channels per socket

AMD EPYC 7002 Series ("EPYC2")

AMD64 SMP SoC, EPYC family

Multi-Chip Module, 9 dies:
- 1 I/O die, off-chip communications (memory, other sockets, I/O)
- 8 “compute” dies (CCDs)
  - Core Complete (CCX) → 4 cores (8 threads), its own L1-L3 cache hierarchy
  - Core Complete Die (CCD) == 2 CCXs: 8 cores (16 threads) + dedicated infinity fabric link to IO die

64 cores (128 threads), 2 socket, 8 memory channels per socket

More info at:
AMD Documentations

WikiChip, AMD EPYC 7742

WikiChip, EPYC Family

AMD EPYC2 On SUSE’s SLE15.1 Tuning Guide

Joint effort by SUSE and AMD

• How to achieve the best possible performance when running SUSE Linux Enterprise Server on an AMD EPYC2 based platform?
• Covers both “baremetal” and virtualization
• “Optimizing Linux for AMD EPYC™ 7002 Series Processors with SUSE Linux Enterprise 15 SP1”

(Done for SLE12-SP3 AMD first gen. EPYC platforms too here)
“Our” EPYC Processor (7742)
“Our” EPYC Processor (7742)

Each CCX has its own LLC:
- NUMA at the socket level (unlike EPYC1)
- More than 1 (16!!) LLCs per NUMA node (unlike most others)
Tuning == Static Resource Partitioning

Virtualization + Resource partitioning: still makes sense?
  • Server consolidation, as EPYC2 servers are very big
  • Ease/Flexibility of management, deployment, etc.
  • High Availability

What Resources?
  • CPUs
  • Memory
  • Θ (will focus on CPU and memory here)
Host vs Guest(s)

Leave some CPUs and some Memory to the host (Dom0, if on Xen)
- For “host stuff” (remote access, libvirt, monitoring, ...)
- For I/O (e.g., IOThreads)

Recommendations:
- At least 1 core per socket
  - Better, if possible: 1 CCX (4 cores) ⇒ 1 “LLC domain”
  - What about 1 CCD (8 cores)? Too much?
- RAM, depends. Say ~50GB

### Huge Pages and Auto-NUMA Balancing

**At host level, statically partition:**
- Static, pre-allocated at boot
- No balancing

**Kernel command line:**
- transparent_hugepage=never
- default_hugepagesz=1GB hugepagesz=1GB hugepages=200

**Libvirt:**
- `<memoryBacking>`
  - `<hugepages>`
    - `<page size='1048576' unit='KiB'/>`
  - `<hugepages>`
  - `<nosharepages/>`
  - `</memoryBacking>`

**In guests: workload dependant**

**Kernel command line:**
- numa_balancing=disable

**Live system:**
echo 0 > /proc/sys/kernel/numa_balancing

---


Power Management

For improved consistency/determinism of benchmarks:

- Avoid deep sleep states
- Use `performance` CPUFreq governor

(At host level, of course! :-P )

If saving power is important, re-assess tuning with desired PM configuration

VM Placement: vCPUs

vCPU pinning

• Pin, if possible, to CCDs:
  – VMs will not share Infinity Link to the I/O die
  – EPYC2: up to 14 (or 16) VMs, 16 vCPUs each

• If not, pin to CCXs:
  – VMs will not share L3 caches
  – EPYC2: up to 30 (or 32) VMs, 8 vCPUs each

• At worst, pin at least to Cores
  – VMs share Inf. Link and L3
  – At least VMs will not share L1 and L2 caches

Libvirt:

```
<vcpu placement='static' cpuset='108-127,236-255'>40</vcpu>
<cputune>
  <vcpupin vcpu='0' cpuset='108'/>
  <vcpupin vcpu='1' cpuset='236'/>
  <vcpupin vcpu='2' cpuset='109'/>
  <vcpupin vcpu='3' cpuset='237'/>
  ...
```

VM Placement: Memory

Put the VM in least possible number of NUMA nodes
Pin the memory to NUMA nodes:
  • If the VM spans both nodes
  • If the VM fist on one node

Libvirt:
  <numatune>
    <memory mode='strict' nodeset='0-1'/>
    <memnode cellid='0' mode='strict' nodeset='0'/>
    <memnode cellid='1' mode='strict' nodeset='1'/>
  </numatune>

VM Enlightenment

Give the VMs a (sensible!) virtual NUMA topology

Give the VMs a (sensible!) virtual CPU topology & CPU model

- Not Passthrough? See later...

Libvirt:
<numa>
  <cell id='0' cpus='0-119' memory='104857600' unit='KiB'>
    <distances>
      <sibling id='0' value='10'/>
      <sibling id='1' value='32'/>
    </distances>
  </cell>
...</n
Libvirt:
<cpu mode="host-model" check="partial">  
  <model fallback="allow"/>
  <topology sockets='1' cores='60' threads='2'/>
...</n
AMD Secure Encrypted Virtualization (SEV)

Encrypts memory
  • per-VM keys
  • Completely transparent

Requires setup both at host and guest level:

SUSE AMD SEV Instructions
Libvirt AMD SEV Instructions

Security Mitigations

Meltdown, Spectre, L1TF, MDS, ...

- AMD EPYC2 is immune to most of them
- Impact of mitigations is **rather small**, compared to other platforms

```markdown
itlb_multihit: Not affected
l1tf: Not affected
mds: Not affected
meltdown: Not affected
spec_store_bypass: Mitigation: Speculative Store Bypass disabled via prctl and seccomp
spectre_v1: Mitigation: usercopy/swapgs barriers and __user pointer sanitization
spectre_v2: Mitigation: Full AMD retpoline, IBPB: conditional, IBRS_FW, STIBP: conditional, RSB filling
tsx_async_abort: Not affected
```

Benchmarks: STREAM

Memory intensive benchmark

• Operations on matrices
  a. In one single thread
  b. In multiple threads, with OpenMP

OpenMP

• OMP_PROC_BIND=SPREAD
  OMP_NUM_THREADS=16 or 32 (on baremetal)
• 1 thread per memory channel / 1 thread per LLC (both on baremetal and in VMs)

Benchmarks: STREAM, 1 VM, single thread

With full tuning, we reach the same level of performance we achieved on the host (look at purple and ... what colour is this?)

Benchmarks: STREAM, 1 VM, 30 threads

With full tuning, we reach the same level of performance we achieved on the host (look at purple and ... what colour is this?)

Benchmarks: STREAM, 2 VM, 15 threads (each)

With full tuning
- Performance of the 2 VMs is consistent (look at red and black)
- Cumulative performance of the 2 VMs matches numbers of the host (look at the purples)

Benchmarks: STREAM, 1 VM, with SEV

On EPYC2, the impact of enabling SEV, for this workload, is very small (less than 1%)

Benchmark: NAS-PB

CPU intensive benchmark
  • Fluid-dynamics computational kernel
  • openMPI

Benchmark: NAS-PB, 1 VM

With full tuning, we reach performance close to the one of the host (look at purple and ... Again!?!?)
On EPYC2, the impact of enabling SEV, for this workload, is again **very small** (less than 1%)
Caveat: CPU Model

On SUSE Linux Enterprise 15.1

• EPYC2 CPU Model not available (QEMU/Libvirt versions)
• XPU Model = `host-passthrough` giving “strange” results

```plaintext
Thread(s) per core:  1
Core(s) per socket:  120
Socket(s):  2
NUMA node(s):  2
```

• EPYC (the model for 1st generation EPYC processors) gives more sensible results

```plaintext
Thread(s) per core:  1
Core(s) per socket:  120
Socket(s):  2
NUMA node(s):  2
```

• (On newer distros, you’ll find the EPYC2 model)
Caveat: CPU Model
Always Double Check, Run Benchmarks!

Host-passthrough, on older QEMU, by not correctly exposing threads, was causing bad performance

More STREAM Benchmarks

Rerun of the STREAM benchmarks
• With Mitigations enabled (were disabled for the Tuning Guide)
• (Much!) More VMs
• Varying pinning of VMs

-- Still work in progress --
-- Benchmarks still running --
-- Data Set not complete yet --
STREAM, 1 VM

VM1
240 vCPUs
**STREAM, 1 VM**

Baremetal:

- MB/sec copy: 256221.30
- MB/sec scale: 173231.56
- MB/sec add: 181804.68
- MB/sec triad: 183952.20

We expect ~Baremetal (with full tuning, look at **purple** and **yellow**).
STREAM, 2 VMs

VM1
120 vCPUs

VM1
120 vCPUs
STREAM, 2 VMs

Baremetal:
MB/sec copy 256221.30
MB/sec scale 173231.56
MB/sec add 181804.68
MB/sec triad 183952.20

We expect ~Baremetal/2
MB/sec copy 128110.65
MB/sec scale 86615.78
MB/sec add 90902.34
MB/sec triad 91976.10
STREAM, 4 VMs

- VM1: 120 vCPUs
- VM2: 120 vCPUs
- VM3: 120 vCPUs
- VM4: 120 vCPUs
STREAM, 4 VMs

Baremetal:
MB/sec copy 256221.30
MB/sec scale 173231.56
MB/sec add 181804.68
MB/sec triad 183952.20

We expect Baremetal/4
MB/sec copy 64055.32
MB/sec scale 43307.89
MB/sec add 45451.17
MB/sec triad 45988.05
STREAM, 6 VMs

- VM1: 40 vCPUs
- VM2: 40 vCPUs
- VM3: 40 vCPUs
- VM4: 40 vCPUs
- VM5: 40 vCPUs
- VM6: 40 vCPUs
STREAM, 6 VMs

Baremetal:
MB/sec copy 256221.30
MB/sec scale 173231.56
MB/sec add 181804.68
MB/sec triad 183952.20

We expect ~Baremetal/X
MB/sec copy 42703.55
MB/sec scale 28871.92
MB/sec add 30300.78
MB/sec triad 30658.70
<table>
<thead>
<tr>
<th>VM</th>
<th>vCPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1</td>
<td>24</td>
</tr>
<tr>
<td>VM2</td>
<td>24</td>
</tr>
<tr>
<td>VM3</td>
<td>24</td>
</tr>
<tr>
<td>VM4</td>
<td>24</td>
</tr>
<tr>
<td>VM5</td>
<td>24</td>
</tr>
<tr>
<td>VM6</td>
<td>24</td>
</tr>
<tr>
<td>VM7</td>
<td>24</td>
</tr>
<tr>
<td>VM8</td>
<td>24</td>
</tr>
<tr>
<td>VM9</td>
<td>24</td>
</tr>
<tr>
<td>VM10</td>
<td>24</td>
</tr>
</tbody>
</table>
STREAM, 10 VMs

Baremetal:
MB/sec copy 256221.30
MB/sec scale 173231.56
MB/sec add 181804.68
MB/sec triad 183952.20

We expect ~Baremetal/10
MB/sec copy 25622.13
MB/sec scale 17323.15
MB/sec add 18180.46
MB/sec triad 18395.22
STREAM, 14 VMs

<table>
<thead>
<tr>
<th>VM</th>
<th>vCPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1</td>
<td>16</td>
</tr>
<tr>
<td>VM2</td>
<td>16</td>
</tr>
<tr>
<td>VM3</td>
<td>16</td>
</tr>
<tr>
<td>VM4</td>
<td>16</td>
</tr>
<tr>
<td>VM5</td>
<td>16</td>
</tr>
<tr>
<td>VM6</td>
<td>16</td>
</tr>
<tr>
<td>VM7</td>
<td>16</td>
</tr>
<tr>
<td>VM8</td>
<td>16</td>
</tr>
<tr>
<td>VM9</td>
<td>16</td>
</tr>
<tr>
<td>VM10</td>
<td>16</td>
</tr>
<tr>
<td>VM11</td>
<td>16</td>
</tr>
<tr>
<td>VM12</td>
<td>16</td>
</tr>
<tr>
<td>VM13</td>
<td>16</td>
</tr>
<tr>
<td>VM14</td>
<td>16</td>
</tr>
</tbody>
</table>
STREAM, 14 VMs

Baremetal:
MB/sec copy   256221.30
MB/sec scale  173231.56
MB/sec add    181804.68
MB/sec triad  183952.20

We expect ~Baremetal/14
MB/sec copy   18301.52
MB/sec scale  12373.68
MB/sec add    12986.04
MB/sec triad  13139.44
STREAM, 30 VMs
STREAM, 30 VMs

Baremetal:
MB/sec copy  256221.30
MB/sec scale  173231.56
MB/sec add    181804.68
MB/sec triad  183952.20

We expect ~Baremetal/30
MB/sec copy  8540.71
MB/sec scale  5774.38
MB/sec add    6060.15
MB/sec triad  6131.74
Conclusions

• Achieving close to host performance in VMs is possible:
  – (In the analyzed workloads)
  – Via resource partitioning
• With KVM, QEMU and Libvirt, on SUSE Linux Enterprise Server 15.1, we can effectively partition the resources to achieve such result
  – With Xen, lacking virtual topology enlightenment for guests
• AMD EPYC2 processor based platforms (especially as far as memory bandwidth is concerned):
  – Guarantees great scalability
  – Offers Memory Encryption with really low overhead
  – Mitigations for hardware vulnerabilities have limited performance impact
About Myself

• Ph.D on Real-Time Scheduling, **SCHED_DEADLINE**

• 2011, Sr. Software Engineer @ Citrix
  The Xen-Project, hypervisor internals,
  Credit2 scheduler, Xen scheduler maintainer

• 2018, Virtualization Software Engineer @ **SUSE**
  Still Xen, but also KVM, QEMU, Libvirt;
  Scheduling, VM’s virtual topology,
  performance evaluation & tuning

• Spoke at XenSummit, Linux Plumbers, FOSDEM,
  LinuxLab, OSPM, KVM Forum, ...
Questions?

(Figure from FOSDEM 2013... I think)

Farewell, Lars
Backup
Virtual VS. Real

(v)CPU Topology:

- vCPUs wander around among pCPUs:
- The hypervisor scheduler moves them!
  - at time $t_1$ vCPU 1 and vCPU 3 run on pCPUs that are SMT-siblings
  - at time $t_2! = t_1$ ... Not anymore!

Shall the guest have a (virtual) topology?

- Yes... if properly constructed, and ...
- ... if we can “rely” on it
- E.g., if the vCPUs are pinned/have hard affinity
Virtual VS. Real: L3-cache & task wakeups

Cache layout: does it affect guest scheduling (& performance)?

- No Yes!!

```c
void ttwu_queue(p, cpu)
if (cpus_share_cache(spm_processor_id(), cpu)) {
    rq_lock(cpu_rq(cpu))
    ttwu_do_activate(cpu_rq(cpu), p)
    ttwu_do_wakeup(cpu_rq(cpu), p)
    check_preempt_curr(cpu_rq(cpu), p)
    /* If cpu_rq(cpu)->curr higher prio */
    /* no IPI to cpu */
    rq_unlock()
} else {
    ttwu_queue_remote()
    llist_add(cpu_rq(cpu)->wake_list)
    smp_send_reschedule(cpu)
    /* IPI to cpu */
}
```
Virtual VS. Real: L3-cache & task wakeups

Cache layout: does it affect guest scheduling (& performance)?
- No
- Yes!!

- `ttwu_queue(p, cpu)`
- if `cpus_share_cache(spm_processor_id(cpu), cpu))` {
- `rq_lock(cpu)`
- `ttwu_do_activate(cpu_rq(cpu), p)`
- `ttwu_do_wakeup(cpu_rq(cpu), p)`
- `check_preempt_curr(cpu_rq(cpu), p)`
- `/* If cpu_rq(cpu)->curr higher prio */`
- `/* no IPI to cpu */`
- `rq_unlock()`
- } else {
- `ttwu_queue_remote()`
- `llist_add(cpu_rq(cpu)->wake_list)`
- `smp_send_reschedule(cpu)`
- `/* IPI to cpu */`
- }

VM cache layout (before QEMU commit `git:9308401`):
- No L3 cache at all

Always send IPI... TO ANOTHER _virtual_ CPU!

Difference shows!
Virtual VS. Real: L3-cache size!

- STREAM benchmark
- AMD EPYC
- VM (KVM) tuned to match host perf
Virtual VS. Real: L3-cache size!

Why copy lags behind when in VM?
Virtual VS. Real: L3-cache size!

Why copy lags behind when in VM?

- Perf
  - on host we were seeing `PREFETCH` instructions being used
  - In VM, no `PREFETCH`! How so?!?!
Virtual VS. Real: L3-cache size!

- "Let’s just expose to the VM whether vCPUs share an L3, no big deal how big such L3 the VM sees!"
- Not quite:
  - Glibc heuristics for deciding whether or not memcpy uses non-temporal stores and `PREFETCH` instrs.
  - \( \text{ths} = (\text{L3 cache size} / \text{nr. threads sharing it}) + \text{L2 cache size} \)
  - Don’t `PREFETCH` if amount of data mem-copied is smaller than \( \text{ths} \)
- We need to expose the correct cache size to the VM
- (still working on it)