NUMA support

Константин Белоусов
kib@freebsd.org

13 декабря 2013 г.

Revision : 1.10

What is NUMA

Machine Architecture

SMP. Full cache coherency and symmetric access to the system resources from any CPU.

NUMA node

Definition: Processor + (almost always) Memory + (possibly) I/O resources.

Cost of access

Intra-node access must be cheaper than inter-node.

Where

Origin

- IMC on CPU die (x86 AMD first, Intel later).
- Multisocket commodity server.
- Intra-package connects (AMD Magny Cours).

Device affinity

- CPU PCIe slots.
- DMI through CPU -> PCH.

Why Bother

Does it matter ?

Generally, NOT much*

Benchmarks

- Single-socket benchmarks of DRAM freq.
- “Real” NUMA tests

NUMA tests

STREAM benchmark

Copy of array. Linux. All threads bound, memory is allocated with libnuma on specific node.

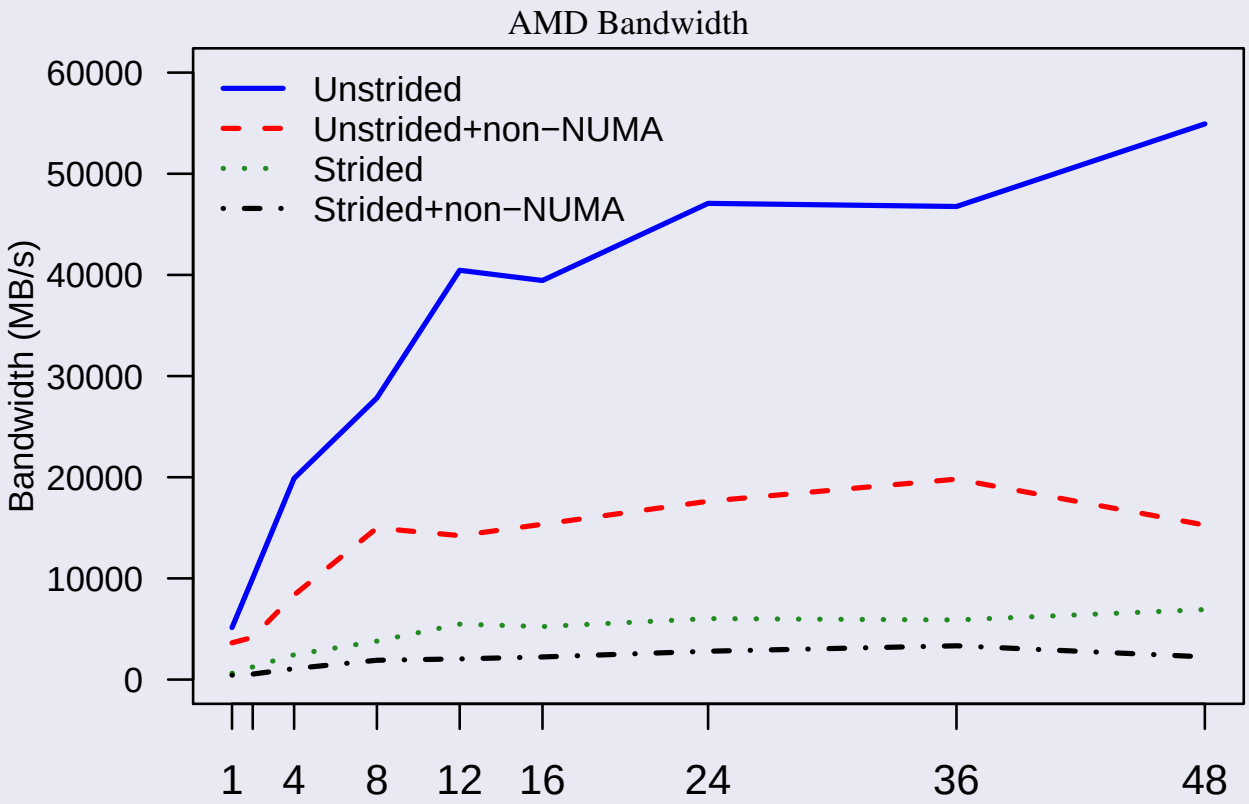
- Unstrided: linear copy, local to node.
- Unstrided+non-NUMA: linear copy on other node.
- Strided: random access to defeat cache and prefetch, local node.
- Strided+non-NUMA: other node.

Reference

Measuring NUMA effects with the STREAM benchmark,
Lars Bergstrom.
`http://arxiv.org/abs/1103.3225`

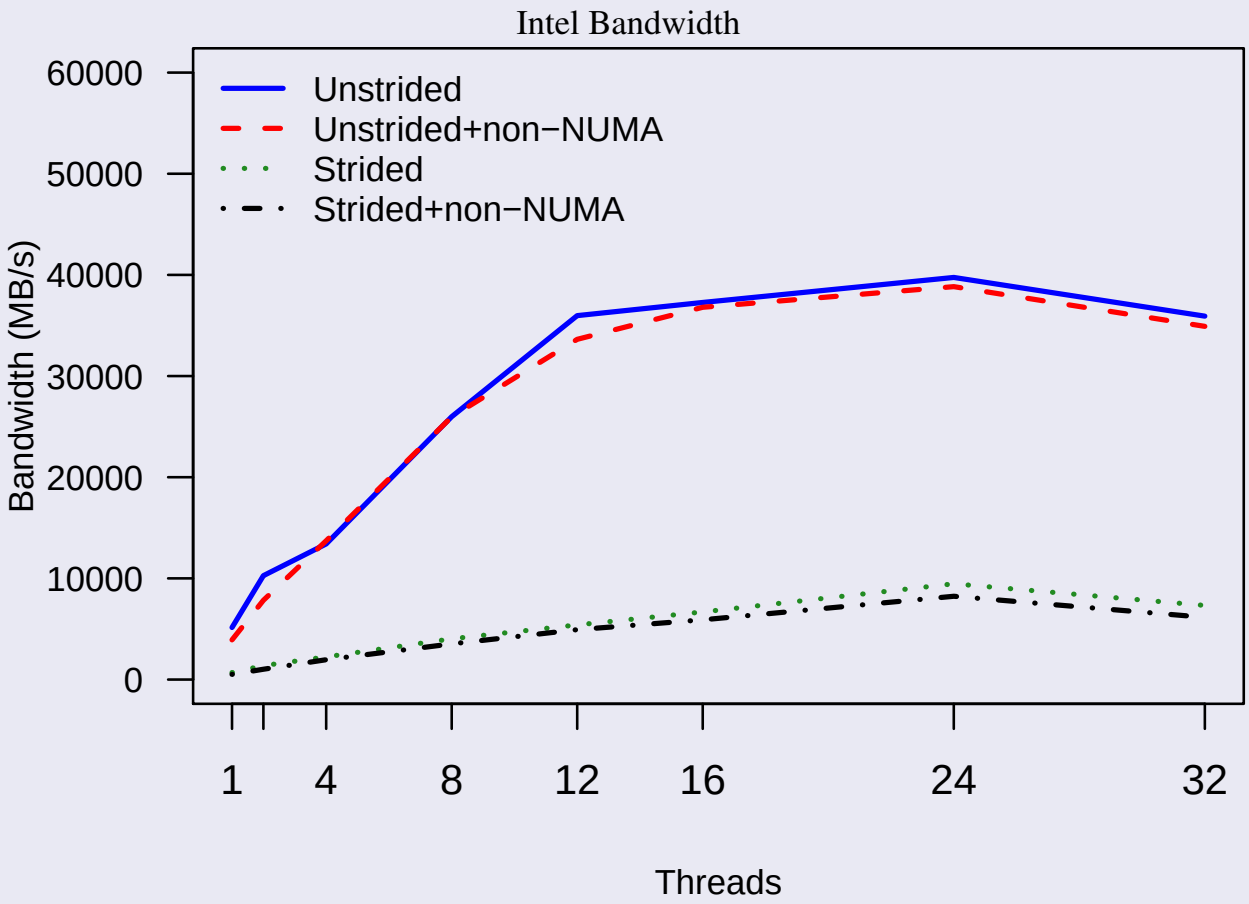
NUMA tests

AMD



NUMA tests

Intel



Discoverability

Memory layout

- BIOS: E820 aka SMAP
- UEFI: GetMemoryMap()

ACPI: SRAT table

System Resource Affinity Table
Maps

- CPUs, identified by APIC id
- Memory ranges (base, length)

to proximity domains. Boot-time only.

ACPI: SRAT table, cont.

SRAT excrept

```
SRAT: Length=960, Revision=1, Checksum=77,
  OEMID=DELL, OEM Table ID=PE_SC3, OEM Revision=0x1,
  Creator ID=DELL, Creator Revision=0x1
  Table Revision=1

  Type=CPU
  Flags={ENABLED}
  APIC ID=0
  Proximity Domain=1

  Type=Memory
  Flags={ENABLED}
  Base Address=0x0000000830000000
  Length=0x0000000800000000
  Proximity Domain=2
```

ACPI, contd.

_PXM method

ACPI Object method, returns proximity domain id for an object.
Works for hot-plug.

SLIT table

System Locality Distance Information
Optional. Cost matrix.

FreeBSD side: Pages and Segments

Page

- Unit of physical memory management.
- `typedef struct vm_page *vm_page_t`.

Physical Segment

- Describes contigous region.
- `struct vm_phys_seg`.
- freelists for phys page allocator.

FreeBSD side: Domains

Domain

- Collection of physical segments.
- `struct vm_domain *vm_phys_domain(vm_page_t m);`

struct vm_domain

```
struct vm_domain {
        struct vm_pagequeue vmd_pagequeues[PQ_COUNT];
        u_int vmd_page_count;
        u_int vmd_free_count;
        long vmd_segs; /* bitmask of the segments */
        boolean_t vmd_oom;
        int vmd_pass; /* local pagedaemon pass */
        struct vm_page vmd_marker; /* marker for pd */
};
```

MT-Pagedaemon

Queues

Each domain has its own:

- Set of page queues: active and inactive.
- Pagedaemon thread: processes the queues.

Page laundry

```
static boolean_t
vm_pageout_launder(struct vm_pagequeue *pq, int tries,
    vm_paddr_t low, vm_paddr_t high);
```

Only laundry domains which contain pages from `[low, high)` range.

MT-Pagedaemon, contd

Coordination

- OOM
- Global paging targets

Current allocator

```
vm_page_t
vm_phys_alloc_pages(int pool, int order)
{
    vm_page_t m;
    int dom, domain, flind;

    for (dom = 0; dom < vm_ndomains; dom++) {
        domain = vm_rr_selectdomain();
        for (flind = 0; flind < vm_nfreelists; flind++) {
            m = vm_phys_alloc_domain_pages(domain, flind,
              pool, order);
            if (m != NULL)
                return (m);
        }
    }
    return (NULL);
}
```

Domain-aware allocator

Jeff patch

- Threading of the requested domain parameter.
- Current thread policy

Stumbling block

Page cache elimination.