Performance Scalability of a Multi-core Web Server

Bryan Veal
Annie Foong

Intel R&D
Overview

• The number of CPU cores on modern servers is increasing rapidly
• Premise: for highly parallel workloads performance should scale with the number of cores
• We tested this premise for web servers
• Our results show that web servers do not scale
• We tested for common problems with poor parallel programming
• We found few parallelism problems in the TCP/IP stack and the web software
• Instead, we found problems inherent to server hardware design
Why Performance Should Scale

- Typical networked servers
  - Have multiple cores
  - Have NICs mapped onto cores
  - Supports many clients
  - Each client has its own flow
- Independence between flows
  - Parallelism in the TCP/IP stack
  - Parallelism the application
- Because of flow-level parallelism, performance should scale
How Performance Scales on Web Servers

Example web server benchmark:
SPECweb2005
- Official results from HP
- Similar scaling for Intel and AMD CPUs
- Performance metric is throughput

• Ideal performance scales linearly
• Actual Performance scales poorly
  - 2x the cores
  - 1.5x the performance

Performance does not scale with the number of cores!
Determining Why Performance Scales Poorly

• Reproduced the published results on our own server
• Tested common causes of poor scaling

• System
  – 8-core Intel Xeon server
  – 4 1GbE NICs

• Software
  – Apache 2, Linux 2.6, PHP 5 Web Server
  – SPECweb2005 Support Workload
    • Highest throughput of 3 SPECweb2005 workloads

• Performance Metrics
  – Compare throughput when increasing from 1 to 8 cores
  – Compare cycles executed per byte transmitted when increasing from 1 to 8 cores
Our Performance Scaling Results

Like the published results, our server scales poorly.
Where Does Cycles per Byte Increase?

OS:Application Ratio is Steady

Either both OS and application are poorly parallelized or something else is affecting them both.
Possible Causes of Poor Scaling

• We investigated many other causes—details in paper

• Potential parallelism problems in software
  – Bad parallelism in the TCP/IP stack
  – Longer code path per flow
  – Stalls due to cache and TLB misses

• Potential scaling problems in hardware
  – Stalls due to system bus saturation
Scaling in the TCP/IP Stack

TCP/IP Stack Throughput

- Removed web server—TCP only
- Bulk transmit
- 6 NICs at line rate
- 128 flows per core

TCP/IP Stack CPU Utilization

- 100%
- 90%
- 80%
- 70%
- 60%
- 50%
- 40%
- 30%
- 20%
- 10%
- 0%

Per-core CPU utilization remains flat

The TCP/IP stack is parallelized well.
Possible Causes of Poor Scaling

• Potential parallelism problems in software
  – Bad parallelism in the TCP/IP stack
  – Longer code path per flow
  – Stalls due to cache and TLB misses

• Potential scaling problems in hardware
  – Stalls due to system bus saturation
Code Path Scaling

- Length of code path may increase with number of cores
- Examples
  - Waiting longer for spin locks
  - Traversing larger data structures
- Increases instructions per cycle (IPC)
- In fact, IPC is decreasing

Code path does not increase significantly.

Decreasing IPC suggests instruction pipeline stalls.
Finding Pipeline Stalls

Top Third Poorest Scaling Functions

- memcpy
- tcp_init_tso_segs
- tcp_ack
- memcpy_c
- free_block
- memset_c
- copy_user_generic_string
- dev_hard_start_xmit
- __alloc_skb
- _zend_mm_alloc_int
- kmem_cache_free
- _zend_hash_quick_add_or_update
- __d_lookup
- ap_merge_per_dir_configs
- skb_clone
- tcp_sendpage
- zend_hash_find

Scaling is harmed the most by stalls for memory loads and stores.
Possible Causes of Poor Scaling

• Potential parallelism problems in software
  – Bad parallelism in the TCP/IP stack
  – Longer code path per flow
  – Stalls due to cache and TLB misses

• Potential scaling problems in hardware
  – Stalls due to system bus saturation
Stalls Caused by Cache and TLB Misses

- More cache and TLB misses can increase memory accesses
- Can be caused by increased data sharing between cores
- In fact, cache and TLB misses are decreasing per cycle

Cache and TLB misses do not cause memory load/store stalls. Something else does.
Possible Causes of Poor Scaling

• Potential parallelism problems in software
  – Bad parallelism in the TCP/IP stack
  – Longer code path per flow
  – Stalls due to cache and TLB misses

• Potential scaling problems in hardware
  – Stalls due to system bus saturation
Possible Cause of Bus Saturation

- The system bus (front-side bus) has two main components
  - **Address Bus** carries requests and responses for data, called **snoops**
  - **Data Bus** carries the data itself

- **Bus Transaction Example**
  - A cache miss generates a snoop on the address bus
  - Snoop is **broadcast** to memory and **all remote caches** to find the most current data
  - Current copy of data is in memory
  - **All remote caches** and memory respond

- More caches mean **more sources and more destinations** for snoops

- **Snoops grow** \( O(n^2) \) **with the number of caches!**
The Effect of Snoops on Scaling

- Snoops may increase bus utilization
- Bus utilization above 2/3 is considered **saturated**
- Data bus utilization increases, but is not saturated
- Confirms data sharing between cores is minimal
- **Address bus utilization increases faster**
- Becomes saturated on 8 cores

**Address bus saturation causes of poor scaling!**
Insights

• Although web servers are highly parallelized and share little data...
• Systems are designed for shared memory applications
• Snoops are broadcast regardless of good parallelism
Conclusions

• Our web server scales poorly with the number of cores
• The OS and application exploit flow-level parallelism and scale well
• Address bus saturation due to broadcast snoops causes poor scaling
Reducing The Effect Broadcast Snoops

• More or faster links between processors (e.g. Intel QuickPath Interconnect, HyperTransport)
  – Because of $O(n^2)$ growth in broadcast snoops
  – Need $O(n^2)$ growth in number or speed of links to keep up
  – Not a scalable solution

• Introduce directories and directory caches
  – Replaces broadcast snooping entirely
  – But comes with more latency and cost

• There is no ideal solution yet—this is a current area of our research.
Thanks!

Questions?