A Case Study of Linux Scalability on Multi-core Platform

ChenYu, Cui Yan
yuchen@tsinghua.edu.cn
ccuuiyyan@gmail.com
Agenda

- Multi-core Challenge
  - Why multi-core?
  - Multi-core applications
- Scalability Definition
- Performance Measurement
  - Benchmarks
  - Performance Tools
- OLTP Scalability on Multi-core Platform
  - OLTP Applications and Hardware Platform
  - TPCC-UVa and Sysbench
  - Performance Analysis
Multi-core Challenge

Why Multi-core?

- Power consumption
- Hard to design more complicated processor (ILP)

Figure 1. Frequency of CPU
Multi-core Challenge

- Moore’s Law in multi-core area

An Intel prediction:
- 2010: 16—64 cores
- 2013: 64—256 cores
- 2016: 256--1024 cores

Figure 2. Moore’s Law
Multi-core Challenge

Applications

Figure 3. Web Search

Figure 4. Science Computing

Figure 5. Mobile computing
Agenda

- Multi-core Challenge
  - Why multi-core?
  - Multi-core applications
- Scalability Definition
- Performance Measurement
  - Benchmarks
  - Performance Tools
- OLTP Scalability on Multi-core Platform
  - OLTP Applications and Hardware Platform
  - TPCC-UVa and Sysbench
  - Performance Analysis
Scalability Definition

- The ability to remain performance when the resource increase
- The ability to increase performance when the core number increase (speedup)

Figure 6. Relationship of HW&SW
Scalability Definition

Bound by some bottlenecks
Agenda

- Multi-core Challenge
  - Why multi-core?
  - Multi-core applications
- Scalability Definition
- Performance Measurement
  - Benchmarks
  - Performance Tools
- OLTP Scalability on Multi-core Platform
  - OLTP Applications and Hardware Platform
  - TPCC-UVa and Sysbench
  - Performance Analysis
Benchmarks

Macro-benchmarks
- Application based or workload based (simulate the real world)
  - TPC-A, TPC-C, TPC-E (OLTP), TPC-H (DSS), TPC-W (e-Commerce)
  - SPECweb2005 (WebServer), SPECMail2008 (Mail Server)

Micro-benchmarks
- Measure micro-operation (e.g. cost of system call, memory read or write bandwidth)
  - Lmbench, STREAM
Performance Tools

- **Linux**
  - top, vmstat, iostat, strace
  - oprofile (cache misses, TLB misses, time-based profile)
  - Systemtap (based on KProbe, kernel profiling)
  - proc (lockstat, scheduler stat)

- **Solaris**
  - Dtrace (script based)

- **FreeBSD**
  - Dtrace (since version 8.0)
Agenda

- **Multi-core Challenge**
  - Why multi-core?
  - Multi-core applications

- **Scalability Definition**

- **Performance Measurement**
  - Benchmarks
  - Performance Tools

- **OLTP Scalability on Multi-core Platform**
  - OLTP Applications and Hardware Platform
  - TPCC-UVa and Sysbench
  - Performance Analysis
OLTP Scalability on Multi-core Platform

- Highly concurrency
- Reasonable respond time
- Update/Insert intensive
- Transaction based
- Database backup

Figure 7. OLTP applications
OLTP Scalability on Multi-core Platform

- Core frequency 1.6G
- L1I cache, L1D cache 32K
- L2 cache 4M
- Cache line 64bytes
- Memory size 4G
- FSB frequency 1.06G
- Hard disk size 500G
- RAID N/A
- Ubuntu 8.04 Linux 2.6.25

Figure 8. Intel hardware platform
OLTP Scalability on Multi-core Platform

- TPCC-UVa
  - Simulate the activity of a company
  - Nine tables in $tpcc$ database: warehouse table, item table, district table, customer table, etc
  - Five kinds of transactions: New Order, Payment, Order-Status, Delivery and Stock-Level
  - Two phases: ramp up, measure
  - Performance: throughput of New Order (tpmC)
Figure 9. TPCC-UVa architecture
OLTP Scalability on Multi-core Platform

- Improvements to TPCC-UVa
  - Single message queue through Unix IPC (message queue) → multi-queue transaction monitor
  - A lightweight load balance module between RTE & TM
  - Benchmark Controller, Checkpoint Controller, Vacuum Controller busy waiting → signal based
OLTP Scalability on Multi-core Platform

- Sysbench-OLTP
  - A modular, cross platform and multi-thread benchmark tool for evaluating OS parameters
  - Two phases: *Prepare* phase and *run* phase
  - Complex transaction, simple transaction and non-transaction
  - Performance of SUT: throughput (tps)
OLTP Scalability on Multi-core Platform

- In TPCC-UVa
  - 90 warehouse and 8 message queue
  - Every warehouse has 10 terminals
  - *Ramp up* time and *measure* time 20 min & 100 min
  - Enable checkpoint controller and disable the vacuum controller
  - The performance close to baseline
  - Profiling by iostat: most of time io-waiting and idle, user time and system time < 1% io-bound
OLTP Scalability on Multi-core Platform

- RAID (Redundant Array of Independent Disks)
  - expensive hardware support
- tmpfs
  - memory backup file system
OLTP Scalability on Multi-core Platform

- How to deal with small memory?
  - 90 warehouse 11G >> 4G
  - every client in RTE: keying time and thinking time
  - \( X(t) \) stochastic process: counting process

\[
P_0 = e^{-\lambda(t-c)} \cdot I_{t\geq c} + I_{0<t<c}
\]

For client: No client request in \( t \)

\[
P = 1 - \left( e^{-\lambda(t-c)} \cdot I_{t\geq c} + I_{0<t<c} \right)^{10 \cdot m}
\]

For message queue: At least a request in \( t \)
## OLTP Scalability on Multi-core Platform

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PostgreSQL</strong></td>
<td></td>
</tr>
<tr>
<td>shared_buffers</td>
<td>3000</td>
</tr>
<tr>
<td>checkpoint_segments</td>
<td>10</td>
</tr>
<tr>
<td>checkpoint_timeout</td>
<td>3600</td>
</tr>
<tr>
<td>max_prepared_transactions</td>
<td>8</td>
</tr>
<tr>
<td>max_connections</td>
<td>100</td>
</tr>
<tr>
<td><strong>MySQL</strong></td>
<td></td>
</tr>
<tr>
<td>innodb_buffer_pool_size</td>
<td>3G</td>
</tr>
<tr>
<td>innodb_log_file_size</td>
<td>750M</td>
</tr>
<tr>
<td>innodb_log_buffer_size</td>
<td>500M</td>
</tr>
<tr>
<td>innodb_flush_log_at_trx_commit</td>
<td>0</td>
</tr>
<tr>
<td>sort_buffer_size</td>
<td>1M</td>
</tr>
<tr>
<td>join_buffer_size</td>
<td>1M</td>
</tr>
<tr>
<td>max_connections</td>
<td>3000</td>
</tr>
<tr>
<td>thread_concurrency</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure 10. database configuration
OLTP Scalability on Multi-core Platform

Figure 11. Scalability Performance

Figure 12. Cache Scalability

Figure 13. TLB Scalability
OLTP Scalability on Multi-core Platform

- Speedup with TPCC-UVa and Sysbench-OLTP 3.8 and 5.2
- Performance breakdown: L2 misses stall contributes most
- TLB effect small
OLTP Scalability on Multi-core Platform

- An universal method to find bottlenecks
  - Degree of Affecting Scalability (DAS)
    - Difference of execution time on Multi-core & Signal core
    - Function related
    - Positive or Negative
  - transaction unit : minimal mixture of transactions
  - Hundreds of functions found
    - TPCC-UVa: 10 functions 52.4%, 50 functions 80.1%
    - Sysbench-OLTP 10 functions 41.5% 50 functions 68.7%
OLTP Scalability on Multi-core Platform

- **Bottlenecks:**
  - FSB (copy_user_generic_string)
    - The most L2 cache misses
    - tmpfs file read (call graph)
    - The cache misses per transaction unit increase with the core number
      - Large work set: experiments (20%)
      - False sharing
    - Cache coherence: data bus, address bus and address bus during the function (26%, 43.7%, 63.7%)
OLTP Scalability on Multi-core Platform

Bottlenecks:

- IPC (ipc_lock)
  - Every kind of IPC resource protected by rw-semaphore
  - Multiple instances of a kind IPC resource operates simultaneously
- Benchmark design
- *Transaction monitor* contention
- lockstat 598 thousand, 383 thousand 80 second test
OLTP Scalability on Multi-core Platform

Bottlenecks:

- Synchronization Primitives
  - In Sysbench-OLTP
    - Synchronization in buffer pool, lock manager and log manager
    - Concurrently read and write, buffer pool, storage hash table for private data and vital data structure
    - Micro-benchmark: futexbench futex_wait and futex_wake
  - In TPCC-UVa
    - LWLockAcquire LWLockRelease
    - WAL and buffer pool
OLTP Scalability on Multi-core Platform

Figure 14. Sysbench execution time

Figure 15. TPCC-UVa execuation time
OLTP Scalability on Multi-core Platform

Figure 16. Context switches per transaction unit
OLTP Scalability on Multi-core Platform

- Bottlenecks:
  - Scheduler
    - Client request $\rightarrow$ SQL statements
    - Server $\rightarrow$ socket (MySQL and PostgreSQL)
    - Lexical analysis, grammar analysis, optimization $\rightarrow$ query result
    - Remote wakeup degrade the scalability (try_to_wake_up)
    - For Sysbench, remote futex wake up
Thanks for your attention!