Multiprocessor Kernel Performance Profiling

Alex Mirgorodskii
mirg@cs.wisc.edu

Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706-1685
USA
Kperfmon: Overview

• Specify a resource
  – Almost any function or basic block in the kernel

• Apply a metric to the resource:
  – Number of entries to a function or basic block
  – Wall clock time, CPU time (virtual time)
  – All Sparc Hardware Counters: cache misses, branch mispredictions, instructions per cycle, ...

• Visualize the metric data in real time
Kperfmon-MP: Goals

Modify uniprocessor Kperfmon to provide:

• Safe operation on SMP machines
  – Thread safety
  – Migration safety

• New feature: Per-CPU performance data
  – More detailed performance data
  – Reduce cache coherence traffic caused by the tool
Kperfmon: Technology

• Use the *KernInst* framework to:
  – Insert measurement code in the kernel at run time
  – Sample accumulated metric values from the user space periodically

• No need for kernel recompilation
  – Works with stock SPARC Solaris 7 kernels
  – Supports both 32-bit and 64-bit kernels

• No need for rebooting
  – Important for 24 x 7 systems
Kperfmon System

In instrumentation request

Kperfmon

Sampling request

Kerninstd

/dev/kerninst

ioctl()

Patch Heap

Data Heap

Kernel Space

Visis
Kperfmon instrumentation

• Counter primitive
  – Number of entries to a function or a basic block

• Wall clock timer primitive
  – Real time spent in a function

• CPU timer primitive
  – Excludes time while the thread was switched-out
  – Can count more than just timer ticks
    • All HW-counter metrics use this mechanism
Non-MP Counter primitive

Code Patch Area

tcp_lookup()

(entry)

Data Area

cnt

sethi hi(&cnt), r0
or r0, lo(&cnt), r0
ldx [r0], r1
retry:
add r1, 1, r2
casx [r0], r1, r2
cmp r1, r2
bne retry
mov r2, r1

Relocated instruction
ba,a tcp_lookup+4

• Atomic, thread-safe update
• Lightweight
• No register save/restore required
Non-MP Wall clock timer primitive

- Inclusive (includes time in callees)
- Keeps accumulating if switched-out
Non-MP CPU timer primitive

- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism
Non-MP CPU timer primitive

- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism
Non-MP CPU timer primitive

- Exclude the time spent while switched out
  - **Instrument context switch routines**
- HW counter metrics are based on this mechanism
Non-MP CPU timer primitive

- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism
Non-MP CPU timer primitive

- Exclude the time spent while switched out
  - **Instrument context switch routines**
- HW counter metrics are based on this mechanism
Non-MP CPU timer primitive

- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism
Non-MP CPU timer primitive

- Exclude the time spent while switched out
  - Instrument context switch routines
- HW counter metrics are based on this mechanism
Non-MP CPU timer primitive

- Exclude the time spent while switched out
  - **Instrument context switch routines**
- HW counter metrics are based on this mechanism

List of paused timers
- head
- free
- free

- **context switch**
  - **tcp_lookup()**: (entry) - **start timer**
  - **switch-out** - **pause timer**
  - **switch-in** - **restart timer**
  - **(exit)** - **stop timer**
Kperfmon-MP: Goals

Modify uniprocessor Kperfmon to provide:

- **Safe operation on SMP machines**
  - Thread safety
  - Migration safety

- **New feature: Per-CPU performance data**
  - More detailed performance data
  - Reduce cache coherence traffic caused by the tool
Thread Safety

Non-MP timer allocation routine

```
ld [head], R1
add R1, 4, R1
st R1, [head]
```

- Used on switch-out to save the paused timers
- Context switch is serial on uniprocessors
  - No thread safety problems there
- Context switches may be concurrent on SMPs!
  - Multiple threads are being scheduled simultaneously
  - The allocation code is no longer safe
Thread Safety

- Context switches may be concurrent on SMPs
- Use the atomic cas instruction to ensure safety
Per-CPU performance data

- Instrumentation code is shared by all CPUs
- Per-CPU copies of the primitive’s data
  - Two copies are never placed in the same cache line

Code Patch Area
- \texttt{tcp\_lookup()}
- (entry)
  - \texttt{...}
  - \texttt{rd cpu\#, r0}
  - \texttt{ldx cnt[r0], r1}
  - \texttt{add r1, 1, r2}
  - \texttt{casx r2, cnt[r0]}
  - \texttt{...}

Data Area
- \texttt{cnt-cpu0}
- \texttt{cnt-cpu1}
- \texttt{cnt-cpu31}
Migration Between Primitives

- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected
Migration Between Primitives

- **Wall timer started on CPU0, stopped on CPU1**
- **Counters and CPU timers are not affected**
Migration Between Primitives

- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected
Migration Between Primitives

- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected
Migration Between Primitives

- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected
Migration Between Primitives

- Wall timer started on CPU0, stopped on CPU1
- Counters and CPU timers are not affected
Solution: virtualization

- Implement wall timers on top of CPU timers!
Solution: virtualization

- Implement wall timers on top of CPU timers!
**Solution: virtualization**

- Implement wall timers on top of CPU timers!
Solution: virtualization

- Implement wall timers on top of CPU timers!
Solution: virtualization

- Implement wall timers on top of CPU timers!
Solution: virtualization

- Implement wall timers on top of CPU timers!
<table>
<thead>
<tr>
<th>Phase: Current Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu0 -</td>
</tr>
<tr>
<td>cpu1 -</td>
</tr>
<tr>
<td>cpu2 -</td>
</tr>
<tr>
<td>cpu3 -</td>
</tr>
<tr>
<td>cpu4 -</td>
</tr>
<tr>
<td>cpu5 -</td>
</tr>
<tr>
<td>cpu6 -</td>
</tr>
<tr>
<td>cpu7 -</td>
</tr>
<tr>
<td>cpu8 -</td>
</tr>
<tr>
<td>cpu9 -</td>
</tr>
<tr>
<td>cpu10 -</td>
</tr>
<tr>
<td>cpu11 -</td>
</tr>
<tr>
<td>cpu12 -</td>
</tr>
<tr>
<td>cpu13 -</td>
</tr>
<tr>
<td>cpu14 -</td>
</tr>
<tr>
<td>cpu15 -</td>
</tr>
<tr>
<td>sum all cpus</td>
</tr>
</tbody>
</table>

entries to (#/sec) 0.0 121.0 242.0
| Phase                  | Branch Mispred | VStall Time | E-$|$ | VMiss Ratio | I-$|$ | VStall Fraction | #/sec |
|-----------------------|----------------|-------------|------|-------------|------|-----------------|-------|
| afs/afs_close/sum all cpus | 9e-06          | 0.005       | 0.008| 2           |
| afs/afs_getpage/sum all cpus | 9e-06          | 0.2         | 0.06 | 1e+01       |
| afs/afs_open/sum all cpus | 9e-06          | 0.2         | 0.07 | 3           |
| genuunix/kmem_alloc/sum all cpus | 0.0001       | 0.3         | 0.1  | 4e+02       |
| tcp/tcp_rput_data/sum all cpus | 7e-06        | 0.1         | 0.2  | 2e+01       |
| ufs/ufs_getpage/sum all cpus | 0.002          | 0.3         | 0.06 | 2e+03       |
| ufs/ufs_lookup/sum all cpus | 0.001          | 0.1         | 0.05 | 2e+03       |
| ufs/ufs_read/sum all cpus  | 0.01           | 0.3         | 0.04 | 2e+03       |
| ufs/ufs_readdir/sum all cpus | 0.0005       | 0.2         | 0.05 | 5e+01       |
Conclusion

• Techniques for correct MP profiling:
  – Atomic memory updates to ensure thread safety
  – Virtualized timers to handle thread migration

• Per-CPU data collection is important
  – Provides detailed performance information
  – Introduces fewer coherence cache misses
Future Work

- New metrics
  - Locality of CPU assignments
  - Per-thread performance data

- Formal verification of instrumentation code for migration/preemption problems

- Ports to other architectures and OS’es
The Big Picture

http://www.cs.wisc.edu/paradyn
The Big Picture

- Demo: Wednesday, Room 6372
- Available for download on request
  - mailto: mirg@cs.wisc.edu
  - Public release in April