Escalonamento Através de Perfilamento em Sistemas Multi-core

Emilio de Camargo Francesquini
emilio@ime.usp.br

Dezembro de 2009

Cenário Atual

- Eterno crescimento da necessidade de poder de processamento
- Dificuldade de projetar processadores cada vez mais rápidos (último exemplo de problemas de projeto: Intel Nehalem Xeon (Core i7/i5) que gera interrupções espúrias [1])
  - O processador Intel Xeon 7400 de 6 cores já possui ~ 2 bilhões de transistores
- Lei de Moore → É possível continuar neste ritmo alucinante de crescimento de desempenho?

Cenário atual – Possíveis Soluções

- Aumento da frequência de funcionamento
  - Aumento linear na frequência → aumento cúbico no gasto de energia
  - Problemas de dissipação de calor
- Otimizações internas ao processador
  - Pipelines mais longos, melhor branch prediction, … → Complexidade de projeto
- Aumento do cache
  - Mais da metade dos transistores dos chips modernos já é dedicada ao cache

Cenário Atual - Soluções

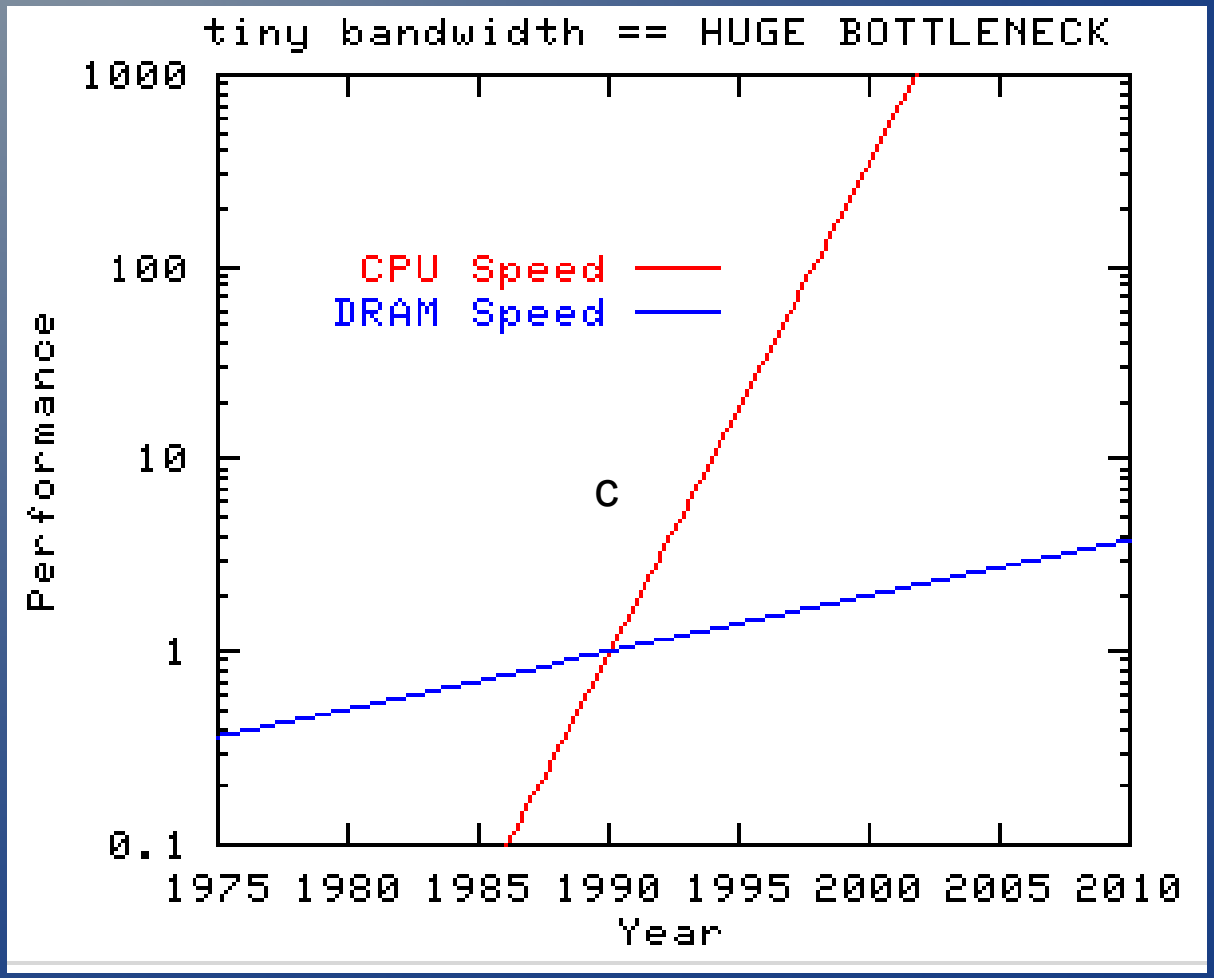
- Nos holofotes → Multi-core
  - Processadores com mais de um core estão a cada dia:
    - Mais baratos
    - Mais presentes
  - 9 entre 10 dos Top 500 (www.top500.org) são muti-core
    - 8 deles com mais de 2 cores
- Por quê?

Cenário atual (cont.)

- Multi-core
  - Simplifica o design dos chips
  - Dá um fôlego novo aos projetistas de hardware
- Mas neste mundo nem tudo é um mar de rosas
  - Memory Wall
  - Programação paralela
  - Utilização eficiente destes processadores
    - Aplicações já existentes

Problemas atuais

- Gargalo de memória



Fonte: www.cs.virginia.edu/stream/

Problemas atuais (cont.)

- Progamação paralela
  - Considerada difícil
  - A maior parte dos programadores não está preparada
  - Possível saída: linguagens funcionais como Scala, Haskell, Erlang, …
- Aumento do número de processos na CPU, banda de memória limitada, cache limitada/cache misses, TLB-misses, ...

Exemplos

- Aplicação simples: 2 threads, um em cada core, incrementando um campo da struct abaixo

```
struct compartilhado {
    unsigned int num_proc1;
    unsigned int num_proc2;
};
```

- Um core invalida o cache do outro a cada operação

Exemplos (cont.)

- Solução 1: colocar os dois threads no mesmo core
- Solução 2: introduzir um deslocamento

```
struct compartilhado {
    unsigned int num_proc1;
    char padding[28];
    unsigned int num_proc2;
};
```

Exemplos (cont.)

- Albuquerque e Hexsel [2] mostraram que o desempenho de um programa paralelo ingênuo de ordenação pelo algoritmo MergeSort pode ser melhorado em pelo menos 12% se forem levadas em consideração as falhas de cache L1/L2 e TLB
- Através do escalonamento com reserva de banda de acesso à memória foi possível mostrar que um processo levou 1,87x mais tempo em dois cores e 3,45x em quatro cores [5]

Soluções atuais

- Definição manual sobre a alocação dos threads/processos aos processadores
  - Máscara de mapeamento processo → processador
  - Definição de um perfil (memory-hungry/CPU-hungry/IO-hungry/...) e distribuição conforme perfil
- Utilização de filas de processamento
  - Apple Grand Central Dispatch

Problemas das soluções atuais

- Exigem conhecimento muito aprofundado (que nem sempre está disponível) da arquitetura alvo
- Falta de portabilidade
- Se o conjunto de processadores é heterogêneo (máquinas NUMA, por exemplo) torna-se impraticável
- O comportamento da aplicação pode mudar com o tempo e com as entradas

Proposta de solução

- Perfilamento do hardware para saber do que ele é capaz
- Perfilamento das aplicações para saber o seu comportamento
- Com posse das duas informações acima e com informações da execução atual e passadas das aplicações determinar um “bom” escalonamento

Exemplos de aplicações de perfilamento de HW

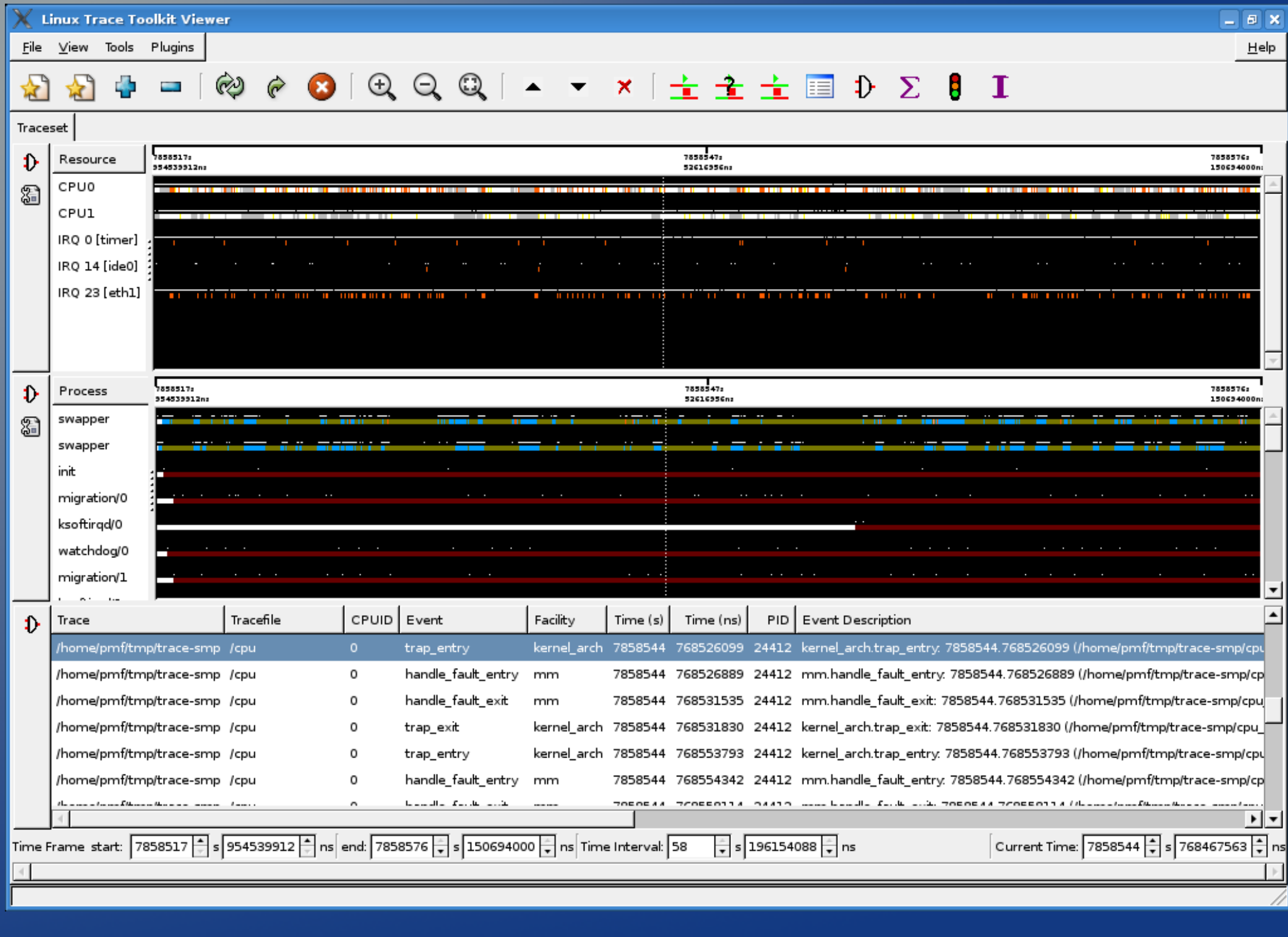
- Stream
  - http://www.streambench.org/
- Portable Hardware Locality - antigo libtopology
  - http://runtime.bordeaux.inria.fr/hwloc/
- Bonnie
  - http://www.textuality.com/bonnie/

Investigando as Aplicações - OProfile

- OProfile
  - utilização dos contadores disponíveis no hardware
    - Falhas/Acertos de L1
    - Falhas/Acertos de L2
    - Falhas/Acertos de TLB
    - Ciclos de processador
    - ...
  - Difícil interpretação dos dados
  - Interferência das demais aplicações e do SO

LTTng

- Inicialmente criado para perfilamento das aplicação em si
- Mais tarde estendido para também levar em conta alguns contadores de hardware [3, 4]
- Conta com interface gráfica bem rica para auxiliar a depuração


Linux Trace Toolkit Viewer
File View Tools Plugins
Help
Traceset
Resource
CPU0
CPU1
IRQ 0 [timer]
IRQ 14 [ide0]
IRQ 23 [eth1]
Process
swapper
swapper
init
migration/0
ksoftirqd/0
watchdog/0
migration/1
Trace Tracefile CPUID Event Facility Time (s) Time (ns) PID Event Description
/home/pmf/tmp/trace-smp /cpu 0 trap_entry kernel_arch 7858544 768526099 24412 kernel_arch.trap_entry: 7858544.768526099 (/home/pmf/tmp/trace-smp/cpu
/home/pmf/tmp/trace-smp /cpu 0 handle_fault_entry mm 7858544 768526889 24412 mm.handle_fault_entry: 7858544.768526889 (/home/pmf/tmp/trace-smp/cp
/home/pmf/tmp/trace-smp /cpu 0 handle_fault_exit mm 7858544 768531535 24412 mm.handle_fault_exit: 7858544.768531535 (/home/pmf/tmp/trace-smp/cpu
/home/pmf/tmp/trace-smp /cpu 0 trap_exit kernel_arch 7858544 768531830 24412 kernel_arch.trap_exit: 7858544.768531830 (/home/pmf/tmp/trace-smp/cpu_
/home/pmf/tmp/trace-smp /cpu 0 trap_entry kernel_arch 7858544 768553793 24412 kernel_arch.trap_entry: 7858544.768553793 (/home/pmf/tmp/trace-smp/cpu
/home/pmf/tmp/trace-smp /cpu 0 handle_fault_entry mm 7858544 768554342 24412 mm.handle_fault_entry: 7858544.768554342 (/home/pmf/tmp/trace-smp/cp
Time Frame start: 7858517 s 954539912 ns end: 7858576 s 150694000 ns Time Interval: 58 s 196154088 ns Current Time: 7858544 s 768467563 ns

Referências

[1] http://www.intel.com/assets/pdf/specupdate/321324.pdf

[2] João Claudio M de Albuquerque e R A Hexsel. OProfile Estendido para Depuração de Desempenho. VI Workshop de Sistemas Operacionais (WSO'09), pgs 1-6, jul 2009

[3] Resource Overbooking and Application Profiling in Shared Hosting Platforms, Bhuvan Urgaonkar, Prashant Shenoy and Timothy Roscoe, ACM Transations on Internet Technologies (TOIT), vol 9, number 1, Pages 1-45, February 2009

[4] Robert W. Wisniewski, Reza Azimi, Mathieu Desnoyers, Maged M. Michael, Jose Moreira, Doron Shiloach, and Livio Soares, Experiences Understanding Performance in a Commercial Scale-Out Environment, *In European Conference on Parallel Computing (*Euro-Par 2007)

[5] Dave Field, Deron Johnson, Don Mize and Robert Stober, Scheduling to Overcome the Multi-Core Memory Bandwidth Bottleneck, Hewlett-Packard Development Company, technical report, November 2007