Energieabschätzung mit Hilfe von Ereigniszählern

Matthias Vogelgesang

7. Februar 2009

Motivation und Problemstellung

- Abschätzung der Leistungsaufnahme von Prozessoren oder Tasks
- Andere Schedulingpolicies möglich
- Charakterisierung der Tasks durch vorhandene Ereigniszähler

Ansatz

- Anzahl der Ereignisse zum Zeitpunkt t_i in Vektor $\mathbf{v}_{t_i}$ zusammenfassen
- A-priori Wissen über Energieverbrauch pro Zählereinheit in $\mathbf{p}$
- Energieverbrauch

$$e = \frac{(\mathbf{v}_{t_i} - \mathbf{v}_{t_{i-1}})^T \cdot \mathbf{p}}{t_i - t_{i-1}}$$

Voraussetzung

- Keine Gleitkommaoperationen im Kern möglich
- Skalierung der Zählgewichte mit 10^3
- 64-Bit-Division mit `do_div` relativ kopfschmerzfrei

Fakten

Sehr moderate Änderungen:

```
diffstat | diff -r -u ~/alt ~/neu
```

```
fs/proc/array.c              |  14 ++++++++++--
include/asm-i386/msr.h       |  19 ++++++++++++++++++++
include/linux/kernel_stat.h  |  14 ++++++++++++++
include/linux/sched.h        |   2 ++
kernel/fork.c                |   3 +++
kernel/sched.c               | 36 ++++++++++++++++++++++++++++++++++++
```

Infrastruktur

- Neues Feld `unsigned long long power` in `task_struct` aus `linux/sched.h`
- Neue Kerndatenstruktur `perf_counter_stat` in `linux/kernel_stat.h`
  - idealerweise pro CPU/Kern
  - tatsächlich statische Variable in `sched.c`

```
struct perf_counter_stat {
  unsigned long long ucycles;
  unsigned long long mlr;
  unsigned long long ldmiss;
  ...
  unsigned long long tsc;
};
```

Auslesen der MSRs

- Auslesen des jeweiligen Registers *r*
- Differenz bilden und mit Gewichten skalieren
- Multiplikation mit 100 damit Raum für Ganzzahldivision

```
#define ADD_POWER(r, stat, scale) \
    rdmsrll(r, msr); \
    (prev)->power += (msr - stat) * scale * 100; \
    stat = msr;
```

Abrechnung in `schedule()`

- Accounting für letzte Task durchführen
- Gewichtete Zählerwerte akkumulieren ($10^3 \cdot 10^2 \cdot nJ$)
- Durch die Zeit (ns) teilen ($\Rightarrow 10^5 W$)

```
rtime = now - prev->timestamp;
if (rtime != 0) {
    prev->power = 0;
    ADD_POWER(MSR_IA32_TSC, pcstat->tsc, 8140)
    ADD_POWER(MSR_P4_BPU_PERFCTR0, pcstat->ucycles, 11500)
    ...
    do_div(prev->power, (unsigned int) rtime);
}
```

Probleme

- Schlechte Übereinstimmung der geschätzten mit gemessenen Werten
  - Allerdings deutliche Unterschiede bei „Microbenchmarks“ zu erkennen
- Keine Berücksichtigung der Grundlast
- `/proc/*/stat` zum Samplen suboptimal