AKBP™/ES© II Team 3

Power/Performance effects of server throttling

Simon Kellner

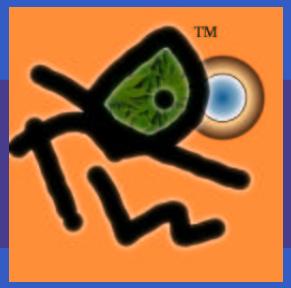
Alexander von Gernler

sisikell@cip.informatik.uni-erlangen.de

sialgern@cip.informatik.uni-erlangen.de

FAU Erlangen-Nürnberg

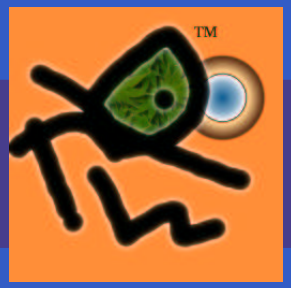
Problemstellung



Netzwerk Server

- interne und externe Clients (Verbindungen)
- interne Verbindungen sollen bevorzugt behandelt werden
- USV-Betrieb erfordert geringe Leistungsaufnahme

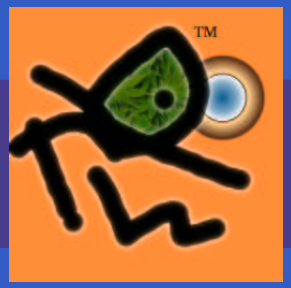
Gewünschte Lösung



Energy-aware Scheduler

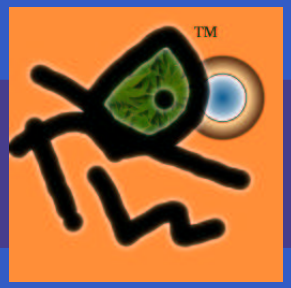
- Netzverbindungen sollen energietechnisch erfasst werden
- Kernel soll Art der Verbindung unterscheiden können
- verschiedene Verbindungsklassen sollen auch unterschiedlich behandelt werden
- Gesamtsystem soll gedrosselt werden können

Verfügbare Ausrüstung



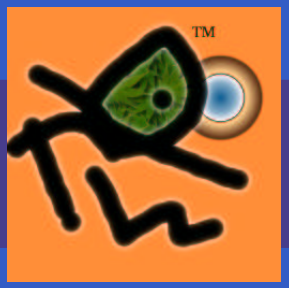
- 2 PCs und eine SunRay pro Team
- serielle Verbindung zwischen den PCs zum Debuggen
- selbstgebastelte™ Energie-Meßeinrichtung
- kgdb debugger stub im Kernel des Zielrechners
- hp kernel patch: loadable scheduler modules

Unser Vorgehen



- Erstellen von Routinen, die Timestamp Counter und Performance Counter lesen
- Erstellen von Testprogrammen, die bestimmte Situationen simulieren:
  - arithmetische Operationen
  - L1-, L2- und Random Memory Zugriffe
  - push/pop Operationen

Testprogramme



arith.c

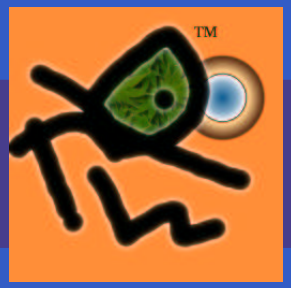
```
for (i=0; i < TH_NUM_COMPUTE; i++) {
    a = ((a<<5)|((a|0x4572feed)&((~a)^0xdeadbeef)))&0x12345678;
    a = (a >> 7);
}
```

cachememtest.c

```
unsigned int *working_set, *tmp;

for (i=0; i < TH_NUM_COMPUTE; i++) {
    *tmp = 0xdeadbeef;
    tmp = working_set + (random() % TH_WORKING_SIZE / sizeof(int));
    a = *tmp;
    tmp = working_set + (random() % TH_WORKING_SIZE / sizeof(int));
    a = *tmp;
    tmp = working_set + (random() % TH_WORKING_SIZE / sizeof(int));
}
```

Testprogramme (2)

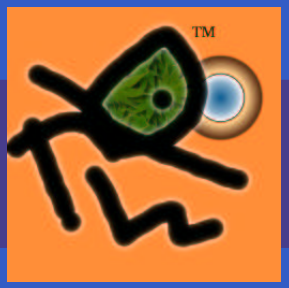


pushpop.c

```
for(i=0; i < TH_NUM_COMPUTE; i++) {
    __asm__("push %eax\n\t"
        "push %ecx\n\t"
        "push %eax\n\t"
        "pop %edx\n\t"
        "pop %ecx\n\t"
        "pop %eax\n\t" );
}
```

Alle Programme zusammen ergeben später den Throttle-Daemon™.

Energiemessungen

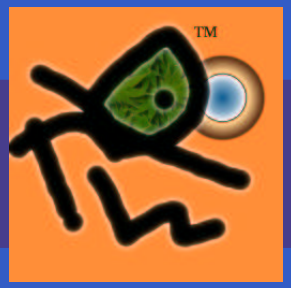


~~Making real tests was too annoying, so we just invent~~

Basierend auf unseren Testprogrammen bekamen wir folgende Ergebnisse:

Testart	P_{CPU}	P_{MEM}	$\sum P$
idle	8	13	21
arith	28	13	41
l1 (seq)	32	13	45
l2 (seq)	32.5	13	45.5
mem(seq)	28	18	46
pushpop	39	13	52
l1 (rand)	36	13	49
l2 (rand)	36.5	13	49.5
mem (rand)	34	18	52

Ergebnisse

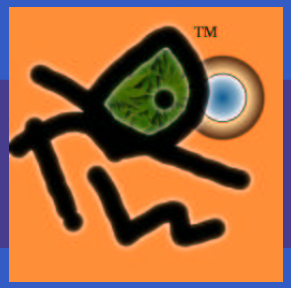


Die Messungen geben Aufschluss über den Energiebedarf gewisser Ereignisse. Nun werden die Ereignisse per Performance Counter erfasst:

- `CPU_CLK_UNHALTED`
- `DATA_MEM_REFS`
- `BUS_TRANS_MEM`

Die Beziehung zwischen Performance Counter und Energie wird im Scheduler ausgenützt.

Energieaccounting

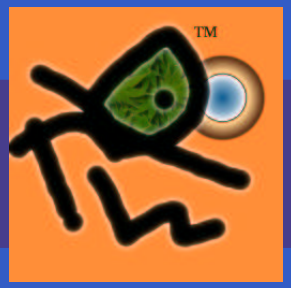


- Performance Counter und TimeStamp Counter ergeben Energiewert
- Erfassung der verbrauchten Energie der letzten ca. 100ms in einer Struktur folgender Art:

0	max_energy	energy_sum

- Modifizierter Scheduler zur globalen Drosselung des Rechners durch `halt()`

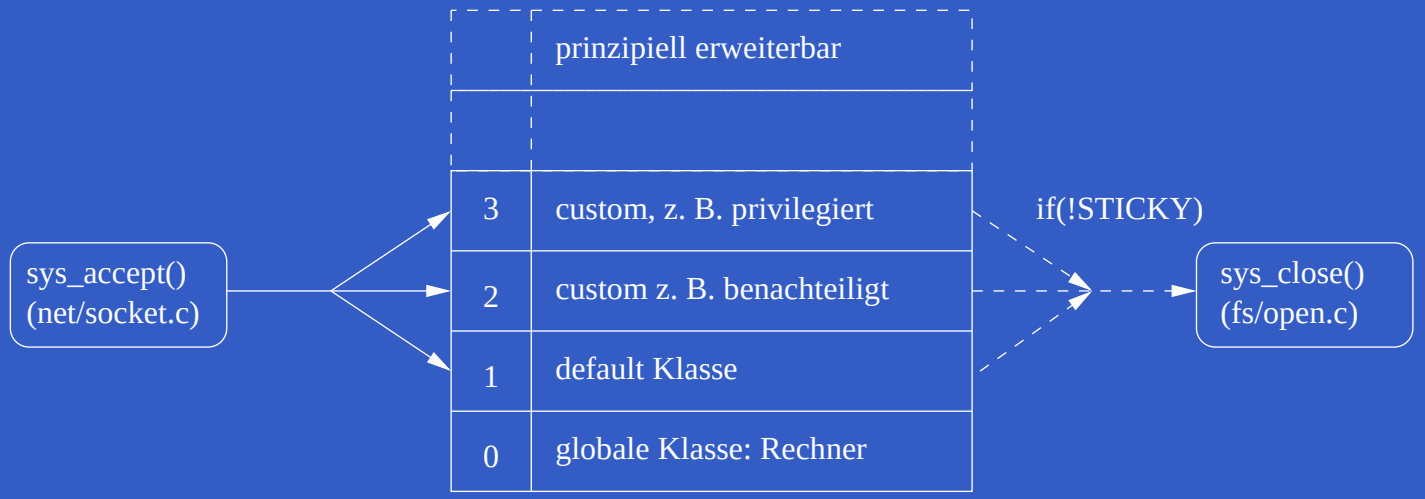
Energieaccounting(2)



Accounting geschieht in

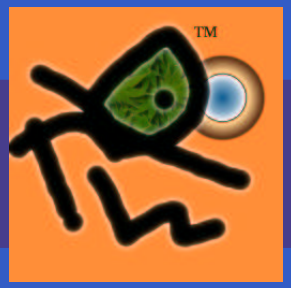
- `choose_next_task()`
  - um CPU zeitnah anhalten zu können
  - für Prozesse mit vielen System-Aufrufen
- `update_one_process()` (Timer-Interrupt)
  - für Langläufer (z.B. unsere Testprogramme, oder `factor`)

Energy Aware Scheduling



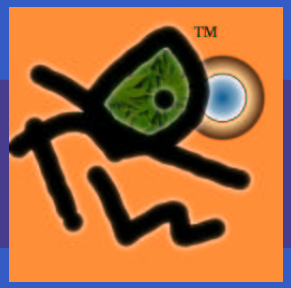
- Hooks in `sys_accept()` und `sys_close()`
- Einteilung der Tasks in Scheduling-Klassen, je nach Source-Adresse

Scheduling Klassen



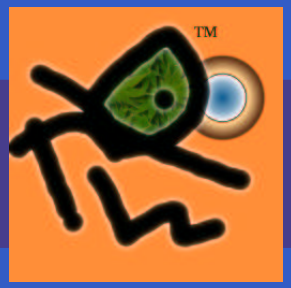
0	Globale Klasse:	Hier kann der Rechner als Ganzes gedrosselt werden
1	Default Klasse:	Jeder Prozess fällt hier hinein, wenn nicht anders bestimmt
x	Custom Klassen:	Geplant: Benutzerdefinierbar über `/proc`

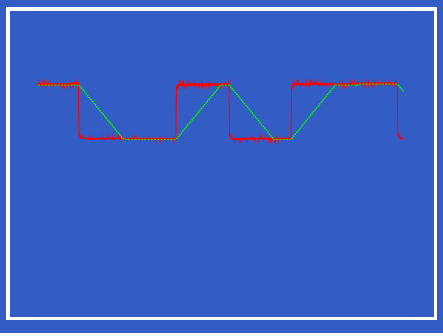
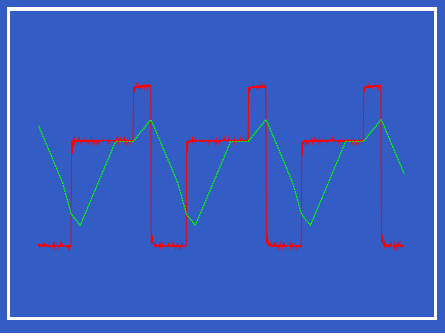
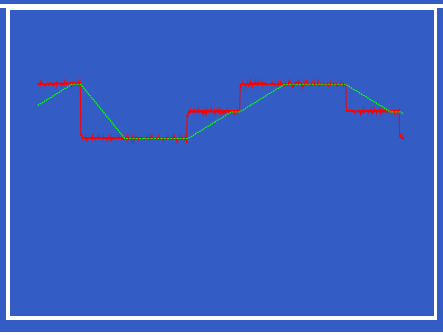
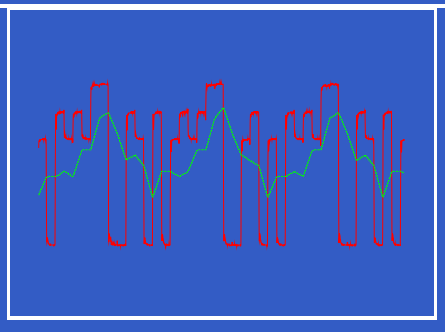
Der Scheduler



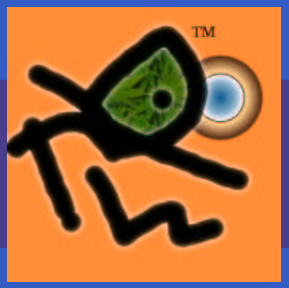
- Energieaccounting
- evtl. kurzes Anhalten des Rechners
- suchen nach attraktivster Task:
  - in der Energieklasse mit höchster Restenergie
  - innerhalb dieser Klasse durch Rest-Timeslices
- evtl. Update der Timeslices
- attraktivste Task (oder idle-Task) wird nächster Prozess

Messwerte unseres Schedulers



	Linux Scheduler	Energy Scheduler
2 Prozesse		
3 Prozesse		

Testanwendung: throttled™

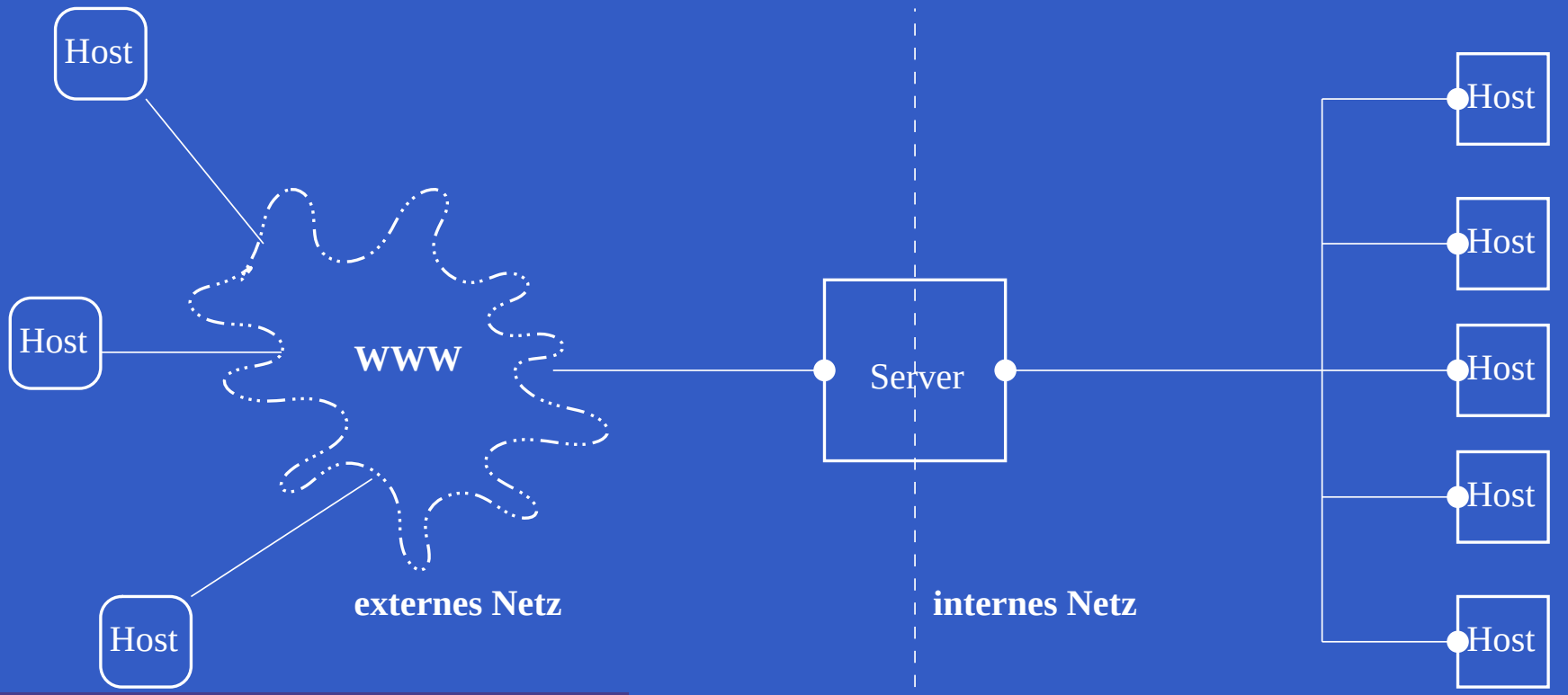


- TCP-Server, lauscht auf definiertem Port
- Daemon forkt für jede neue Verbindung
- Führt pro Verbindung Berechnungen durch

```
team3/sisikell> telnet throttle 22914
Trying 192.168.34.33...
Connected to throttle.
Escape character is '^]'.
Spawned child on pid 923, this is hashed to 920.
Starting calculation loop:
0..10..20..30..40..50..60..70..80..90..100
Calculation loop finished.
Connection closed by foreign host.
team3/sisikell>
```

Testszenario

- 1 privilegierter, 1 normaler Host
- privilegierter ist deutlich schneller fertig



Noch Fragen, Kienzle?