Symmetrisches Multiprocessing mit Linux

Michael Engel
Unix-AG, Universität-GH Siegen
engel@unix-ag.org

27.06.98

Zusammenfassung

SMP erreicht durch die Verfügbarkeit preiswerter Multiprozessor-Systeme den Normalanwender. Zukünftig wird die Bedeutung von SMP zur Leistungssteigerung von Systemen wachsen, da die Geschwindigkeit von Prozessoren physikalische Grenzen erreicht. Der Einsatz von SMP unter Linux bietet für viele Probleme interessante Lösungsansätze.

Überblick

- SMP und andere Ansätze zur Parallelisierung
- Probleme von SMP
- Entwicklung von SMP
- Plattformen
- Compilieren für und mit SMP
- Multithreading und SMP
- Performance
- Ausblick

SMP und andere Ansätze zur Parallelisierung

- **Shared Memory MP** - symmetrische Multiprozessorkonfiguration, alle Prozessoren haben Zugriff auf den gesamten Speicher und alle Geräte
- **Shared Disk MP** - Jeder Prozessor hat eigenen lokalen Speicher, nur Festplatten werden gemeinsam genutzt
- **Shared Nothing MP, Clustering** - durch LAN oder WAN verbundene, unabhängige Systeme, Kommunikation durch spezielle Libraries wie PVM oder MPI (*Beowulf-Cluster* der NASA)

Vorzüge und Nachteile von SMP (1)

Einer der wichtigsten Aspekte bei Multiprozessorsystemen ist die Skalierbarkeit. Diese ist abhängig von der Aufgabenstellung – einige Probleme sind besser skalierbar als andere. Zudem ist die Skalierbarkeit von der Art des implementierten Multiprocessings abhängig.

Vorzüge von SMP:

- System erscheint für jeden Prozeß wie ein Einprozessorsystem
- keine Programmanpassungen vorhandener Software an System notwendig, Vorteile von SMP können durch Anpasung der Software bei Bedarf dennoch genutzt werden.
- durch die gemeinsame Verwendung von Hauptspeicher und Peripheriegeräten sind SMP-Systeme recht preiswert realisierbar.

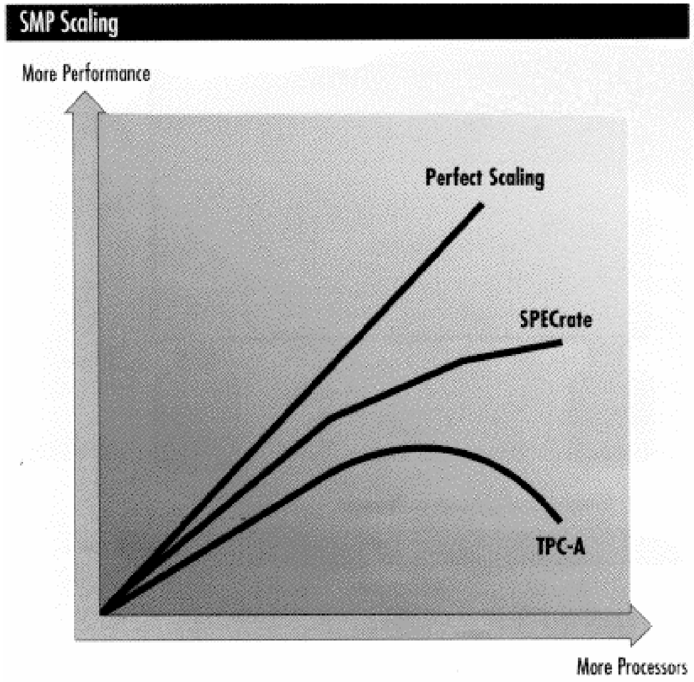
Vorzüge und Nachteile von SMP (2)

Nachteile von SMP:

- Realisierung der Einprozessor-Illusion aufwendig für Hardware und Betriebssystem
- Hohe Auslastung des Busses oder Switches, der die Prozessoren mit dem shared memory verbindet
- Wettbewerb um gemeinsam benutze Speicherbereiche - Locks müssen den Zugriff auf Speicherbereiche regeln $\rightarrow$ Aufwand für das BS

Die meisten der im kommerziellen Bereich eingesetzten Systeme sind shared memory SMP-Systeme.

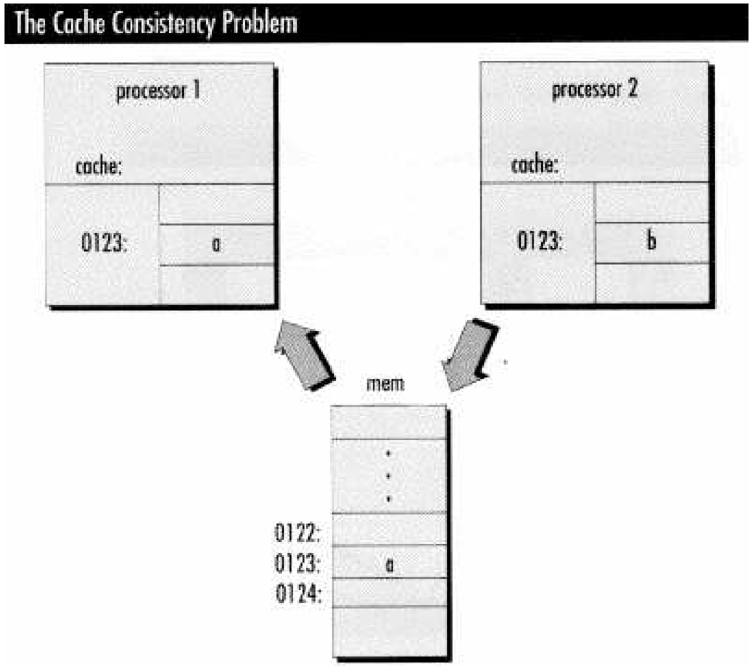
Skalierbarkeit von SMP



Skalierbarkeit von MP bei verschiedenen Benchmarks

Das Cache-Konsistenzproblem (1)

Eines der grundlegende Probleme von SMP-Systemen ist die Cache-Konsistenz.

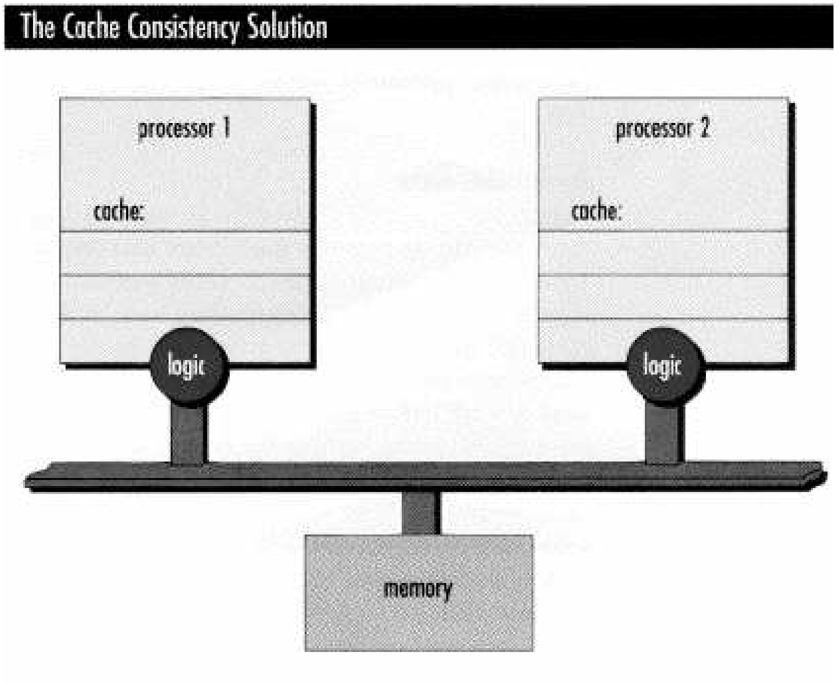


Das Cache-Konsistenzproblem (2)

Lösung des Cache-Konsistenzproblems:

- Logik an jedem Prozessor-Bus-Interface meldet Änderungen des lokalen Cache-Inhaltes an andere Prozessoren
- Logik reagiert auf Meldungen der anderen Prozessoren und invalidiert evtl. ungültige Einträge im lokalen Cache
- Gegenseitige Invalidierung erhöht die Anzahl der cache misses, Überwachungsprotokoll trägt zusätzlich zur Erhöhung des Verkehrs auf dem Bus bei $\rightarrow$ Reduzierung der Leistung und Skalierbarkeit

Das Cache-Konsistenzproblem (3)



Locks (1)

Eines der weithin bekannten Probleme von SMP ist die Aufrechterhaltung der Datenkonsistenz. Betrachten wir das klassische Beispiel einer doppelt verketteten Liste. Entfernen eines Elementes aus der Liste erfordert, daß zwei Zeiger aktualisiert werden:

- Der Vorwärtszeiger des dem zu löschenden Element vorhergehenden Elementes
- Der Rückwärtszeiger des Elementes, das auf das zu löschende Element folgt

Wenn nur einer der beiden Zeiger aktualisiert wurde, befindet sich die Liste in einem inkonsistenten Zustand. Wenn zwei Prozessoren in einem Multiprozessorsystem gleichzeitig Elemente aus der Liste entfernen oder zu ihr hinzufügen, kann das Ergebnis unvorhersehbar sein.

Locks (2)

Ein kritischer Abschnitt ist ein Abschnitt von Code, der gemeinsam genutzte Daten modifiziert und deshalb nicht von mehr als einem Proze gleichzeitig ausgeführt werden darf. Ein Prozeß mußal so warten bis der andere Prozeß seinen kritischen Abschnitt beendet hat. Die beiden kritischen Abschnitte müssen also serialisiert werden.

Die Lösung für dieses Problem sind Locks.

Prinzipiell ist ein Lock einfach ein Bit im Speicher, das von Prozessen genutzt wird, um Zugang zu kritischen Abschnitten zu regeln. Wenn ein Prozeß einen kritischen Abschnitt betreten will, prüft er den entsprechenden Lock. Wenn das Bit aus ist, schaltet der Prozeß es an und betritt dann den kritischen Abschnitt. Wenn das Bit an ist, wartet der Prozeß bis das Bit aus ist. Jeder Prozeß muß das Bit beim Verlassen des kritischen Abschnittes zurücksetzen.

Locks (3)

Spinlocks vs. Blockierende Locks

Wenn ein Prozeß einen Lock haben möchte, der bereits von einem anderen Prozeß gehalten wird, muß er warten. Ein **spin lock** erlaubt es dem wartenden Prozeß den Prozessor zu behalten, wobei wiederholt die Prüfung des Bits in einer kleinen Schleife (spin) solange, bis der Lock vefügbar wird.

Spin locks sind nützlich für Locks, die nur für sehr kurze Zeit gehalten werden. Ein **blockierender Lock** hält den Prozeß so lange an, bis der Lock frei ist und stellt in dann in die run queue zurück. Dies ist sinnvoll bei Locks, die für längere Zeit gehalten werden sollen.

$\rightarrow$ Warten vermindert immer die Systemleistung, egal wie es implementiert ist.

Entwicklung von SMP (1)

Erste Multiprozessor-Systeme kamen in den 60er Jahren auf den Markt. Eine frühe SMP-Maschine ist die Burroughs 5500, die direkt für die Unterstützung von Hochsprachen (Algol) ausgelegt war.

UNIX-Support für SMP-Systeme kam Mitte der 80er Jahre mit Systemen von Data General (AViiON) und Sun (z.B. SparcServer 670) auf. Diese waren zunächst noch als Highend Server-Systeme konzipiert.

Erste Versionen von Linux/SMP wurden 1995 von Alan Cox et al. unterstützt.

Benutzbar wurde der SMP-Support in Linux etwa seit der Kernelversion 2.0.4, aktuelle 2.0.x-Kernel und 2.1.x-Kernel sind SMP-fähig.

Plattformen

Linux/SMP ist auf folgenden Plattformen verfügbar:

- intel Pentium/PentiumPro/Pentium II MPS 1.1/1.4-konforme Systeme
- Multiprozessor SparcStations, z.B. Sparc 10, Sparc 20

Linux/SMP befindet sich für diese Plattformen in Entwicklung:

- Digital Alpha (erster Support ab Kernel 2.1.92)
- Motorola/IBM PowerPC

SMP bei intel-Architekturen (1)

- Erste Multiprozessor 80386-Systeme 1987 von **Sequent** – *Sequent Symmetry*
- Einführung der SMP-Standards **MPS 1.1** und **MPS 1.4** (1994) durch intel
- Seit 1995 Multiprozessor-Boards für Pentium-Prozessoren verfügbar
- Heute hauptsächlich Einsatz von PentiumPro und Pentium II-CPUs

Intel-kompatible CPU's (AMD K6, Cyrix 6x86) verwenden ein differentes Multiprocessing-Protokoll, da MP 1.1 und 1.4 von intel patentiert sind. Bisher existiert leider noch kein Chipsatz, der diese Spezifikation unterstützt.

SMP bei intel-Architekturen (2)

Bei intel-Systemen sind architekturbedingt Einschränkungen in Kauf zu nehmen:

- Pentium: Alle CPU's teilen sich üblicherweise einen Level-2-Cache
- Pentium II: Maximal 2 CPU's je System, max. 512 MB cacheable Area

Aus SMP-Sicht ist der ideale intel-Prozessor der PentiumPro. Systeme mit bis zu 4 Prozessoren sind vorgesehen, mit „Tricks“ bis zu 8 CPU's möglich (ALR etc.).

SMP auf SPARC-Systemen

Sun vertreibt seit einigen Jahren SPARC-basierte Multiprozessorsysteme. Das erste Desktop-System war die SparcStation 10, Nachfolgemodelle sind z.B. die SparcStation 20, UltraSparc 2 sowie diverse Enterprise-Server von Sun. Einige ältere Sparc-Prozessormodule (SM30/40) sind nicht multiprozessor- fähig.

Prinzipiell unterstützt Linux auf SPARC alle Konfigurationen, die auch von Solaris 2 unterstützt werden. Eine Ausnahme bilden Systeme, die noch auf der Sun 4-Architektur basieren, z.B. SparcServer 670. Diese Systeme werden allerdings auch von Solaris nur bis zur Version 2.4 unterstützt. Desktop-Workstations unterstützen normalerweise bis zu 4 CPUs, Enterprise- Server sind mit bis zu 64 Prozessoren erhältlich.

Viele der „größeren“ Systeme sind allerdings mangels Verfügbarkeit eines Systems bei den Entwicklern noch nicht getestet.

Compilieren eines SMP-Kernels

Die einzigen Änderungen sind in `/usr/src/linux/Makefile` vorzunehmen:

```
# NOTE! SMP is experimental. See the file Documentatio
#
SMP = 1
#
# SMP profiling options
SMP_PROF = 1
```

Compiliert wird der Kernel dann mit

```
% make dep clean
% make -j4 zImage modules modules_install
```

SMP-Kernel

Beim Konfigurieren eines Kernels für ein SMP-System sollte noch folgendes beachtet werden:

- Die Benutzung der Option -j ohne Parameter startet so viele Prozeße wie möglich. Bei Systemen mit wenig Speicher kann ein Compilierungsvorgang mangels ausreichender Resourcen abbrechen.
- Linux hat auf SMP-Systemen noch ein Problem mit Zeitdrift, d.h. die Systemzeit entfernt sich während des Betriebs von der Echtzeit.
  Abhilfe bringt hier Zeitsynchronisation via Netzwerk mit xntpd oder der Einsatz eines DCF77-Zeitsignalempfängers.

Dateien in `/proc` (1)

Ein Kernel mit aktivierter SMP-Unterstützung führt detailliertere Informationen in einigen der Pseudodateien in `proc` auf:

`/proc/cpuinfo`

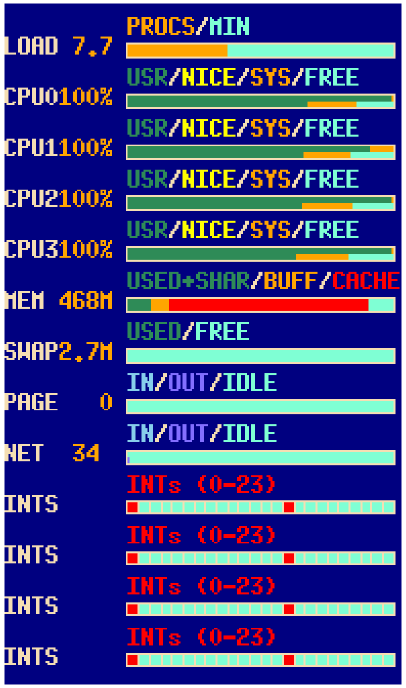
```
processor       : 0
cpu family      : 5
model           : Pentium 75+
stepping        : 12
[...]
processor       : 1
cpu family      : 5
model           : Pentium 75+
stepping        : 12
```

Dateien in `/proc` (2)

`/proc/interrupts`

```
           CPU0       CPU1
  0:    1193305        395    XT-PIC           timer
  1:          1          1   IO-APIC-edge      keyboard
  2:          0          0    XT-PIC           cascade
  4:          6         18   IO-APIC-edge      serial
  5:       1524       1196   IO-APIC-edge      soundblaste
  7:     908092     908114   IO-APIC-edge      3c509
 13:          4          0    XT-PIC           fpu
 17:      19391      19353   IO-APIC-level     ncr53c8xx
NMI:          0
IPI:          0
```

xosview



xosview ist ein X11-basiertes Tools, um CPU- und Systemauslastung grafisch anzuzeigen. Schön zu sehen ist neben der CPU- auch die Interrupt- Auslastung und Speicherbelegung.

Ausnutzung von SMP

Die Vorteile von SMP lassen sich unter folgenden Bedingungen nutzen:

- Viele Benutzer haben Zugriff auf das System
- Aufgabenstellungen bestehen aus vielen separaten Prozessen (z.B. Compilieren eines großen Projektes)
- Applikationen können durch Ausnutzung von Multithreading mehrere Prozessoren nutzen

Multithreading und SMP

In Situationen, in denen eine Aufspaltung eines Problems in mehrere unabhängige Prozesse schwierig ist, empfiehlt sich die Verwendung einer Threading-Library, die SMP unterstützt.

Xavier Leroy's LinuxThreads ist eine recht komplette und solide Implementation des POSIX 1003.1c Threading-Standards. Dabei wird - im Gegensatz zu den meisten POSIX-Thread Portierungen - der clone()-Systemaufruf des Linux-Kernels verwendet.

LinuxThreads ist definitiv das Paket, das zur Entwicklung großer Programmpakete eingesetzt werden sollte.

Linux SMP-Performance

Die folgenden Systeme wurden für einen Performance-Vergleich benutzt:

- Single-CPU PentiumPro 200
- Dual-CPU PentiumPro 200
- Quad-CPU PentiumPro 200

Als Test wurde jeweils gcc-2.7.2.3 und Linux-2.1.102 compiliert.

	gcc	Linux
Single	11:53	
Dual PP200	6:07	
Quad PP200	3:13	

Einschränkungen von Linux/SMP

Momentan kann Linux/SMP noch nicht die volle Performance der Multiprozessor- Systeme ausnutzen. Aus Gründen der Komplexität ist SMP noch so implementiert, daß der Kernel von einem großen Lock geschützt wird. Das bedeutet, daß sich zu jedem Zeitpunkt exakt ein einziger Prozeß im Kernelmodus befinden kann. An der Reduzierung der Lock-Granularität wird momentan gearbeitet; mit aktuellen 2.1.x-Kernelversionen ist bereits eine Performancesteigerung merkbar.

Ein weiteres Problem mit SMP ist, daß einige Treiber noch Probleme mit SMP haben, insbesondere mit der Verteilung der Interrupts auf mehrere CPUs. Die meisten Netzwerktreiber sind dank der Arbeit von Donald Becker mittlerweile SMP-fähig, Probleme bereiten aber z.B. diverse SCSI-Treiber, darunter besonders der für die verbreiteten Adaptec 2940-Controller und den AIC-7880 Chipsatz.