

L 08: Vermischtes

- Deadlocks vs. Virtualisierung
- Nicht OS-orientierte Virtualisierung
- Standards
- Band-Verarbeitung
- Boot Prozess

Fragen

- Wann etwa entstanden OSI und TCP/IP?
- Warum hat OSI mehr Ebenen als TCP/IP?
- OSI Ebenen in TCP/IP, aber nicht normiert? Warum?
- OSI-Schicht für
 - Ende-zu-Ende Kommunikation?
 - Komprimierung und Verschlüsselung anzubieten?
- Welches sind "Basis"-Dienste im TCP/IP?
- Neue Entwicklungen bei Netzwerktreibern?
 - Warum? Lösungen?
 - Vorteil bei VNICs (Xen, Solaris Container)?

Verklemmung (Deadlock)

- Deadlock, wann?
 - Konkurrierender Zugriff auf Ressourcen
 - Kein Ausweg programmiert
- Lösungen:
 - Scheitern bei Nicht-Verfügbarkeit
 - Allokation am Anfang
 - Queues / Region mit definierten Ressourcen

Virtuelle Ressourcen

Moderne OS: Virtualisierung aller Ressourcen

- CPU: Prozesse, Threads
- Memory: Virtual Memory
- Disk: Volume Manager, Filesysteme
- Netzwerk: TCP/IP Stack,
Netzwerk-Virtualisierung
- IO-Geräte: (Karten-Leser,-Stanzer, Drucker, ...)
Spool Queues
(*Simultaneous Peripheral Operation On-Line*)

Deadlock in heutigen OS?

Virtuelle Ressourcen verhindern Deadlocks

Wenige Stellen bleiben übrig:

- Dedizierte CPU Zuteilungen
- Memory-Overcommitment

Achtung: Deadlocks in Applikationen

- Können weiterhin auftreten!
- Das Betriebssystem kann sie nicht verhindern!

Virtualisierung

Virtualisierung ist z.Zt. ein Hype-Begriff

- Virtualisierung des OS Environments
 - VMware, Xen, Parallels, Solaris xVM, Linux KVM
 - BSD Jails, Solaris Zonen, Virtuozzo
 - Grid Cluster
- Virtualisierung des Applikations-Environments
 - API Ebene
 - Architektur

Grid Cluster (1)

Entspricht den Batch-Systemen älterer OS

- Definieren eines Jobs
 - OS + HW-Plattform
 - Bibliotheken
 - Eingabe-Dateien
 - Verfügbarkeit von Storage Ressourcen (z.B. Disk, Tape)

Grid Cluster (2)

- Job Scheduler
 - Meist redundant ausgelegt
 - Berücksichtigt Abhängigkeiten
 - Kann idle-Rechner mit einbeziehen
 - Wenig benutzte Server
 - Unbesetzte Desktops

Beispiele:

- Net Job Entry, ..., LSF
- Load Leveler, Torque/MAUI, Sun Grid Engine

Virtualisierung auf API Ebene

Multiprozessing über Rechner hinweg

- Basis-Framework zum Starten der Prozesse
- Spezielle API zum parallelen Funktionsaufruf
 - explizites fork/join (heterogene Teilaufgaben)
 - iteratives fork (homogene Teilaufgaben)
- Transport der Daten via Framework
- Transport des Ergebnisses via Framework

Beispiele: MPI, PVM (Quasi-Standards)

Virtualisierung der Architektur

Neue Aufteilung der Applikation

- Kein zentrales Programm mehr
- Services, die Daten-Objekte bearbeiten/erzeugen
- Definition von Datenfluss und Interfaces
- SOA (Service Oriented Architecture)
- Automatische Verteilung auf viele Rechner
- ggf. inklusive hoher Verfügbarkeit

Erfordert eine neue Generation von Applikationen

- Umdenken aller Beteiligten notwendig

Standards

Wozu braucht man Standards?

Standards, wozu?

- Langlebigkeit von Applikationen
- Datensicherheit
- Reproduzierbarkeit
- Revisions-Anforderungen
- Portierbarkeit auf andere Systeme

Standards: POSIX

- **Basis-Definitionen** **POSIX.1/IEEE 1003.1-2001)**
 - Konventionen, Definitionen
 - C-Headerdateien
- **Shell und Tools** **POSIX.2/IEEE 1003.1-2001)**
 - ksh ist Standard
 - Tools wie awk, sed,
- **Realtime** **POSIX.4/IEEE 1003.1b-d**
- **Threads** **POSIX.4a/IEEE 1003.1c**
- **System-Schnittstelle: (Funktionen in C-Bibliothek)**

POSIX Betriebssysteme

POSIX compliant:

A/UX, AIX, BSD/OS, Darwin (Mac OS X), HP-UX, Irix,
LynxOS, MINIX, OpenVMS, penOS, QNX
Solaris, OpenSolaris, UnixWare, velOSity, VxWorks, ...

Nahezu compliant:

BeOS/Haiku, GNU/Linux (LSB Standard),
FreeBSD, NetBSD, OpenBSD, ...

POSIX Subsystem:

Windows* (Unix Services f. Windows notwendig)
MVS, z/OS, ...

Standards: API Level

Programmierung der Anwendung auf API

- Unabhängigkeit vom OS
- Sehr gute Portierbarkeit

Beispiele:

- Java / JavaFX / Jakarta (Applikationen)
- J2EE (App-Server / Web Applikationen)
- PL/SQL für Oracle Datenbanken
- ABAP für SAP Applikationen

Standards: Graphik

Unabhängiger Standard:

- Network Extensible Window System (früheres Graphik System von Sun)
- X Window System (X, X 11)
- OpenGL (3D Applikationen)

Oberflächen:

- OpenWindows
- Motif (LessTif)
- KDE, Gnome (beide stark evolvierend...)

Firmen Standards

- garantieren nur Aufwärtskompatibilität
- erschweren die Portierung zur Konkurrenz
- Beispiele:
 - Windows mit Graphic API, Visual Basic
 - MacOS, Symbian
 - Mainframe Welt (außerhalb POSIX Subsystem)

Probleme:

- Langlebige Benutzbarkeit von Programmen
- Langlebigkeit von Dokumenten

OS Einbindung Bandlaufwerke

Bandlaufwerke

- Sequentielle IO (Datenblöcke, Gaps)
- Positionierung über *tape mark*
- Blocks verschiedener Größe (vs. Effizienz)
- Hohe Kapazität, Lebensdauer, niedrige Kosten
- Limitierte Anpassung an Datenrate (read/write)
 - Streaming Mode (Geschwindigkeit || Datenrate)
 - Start/Stop Mode (Datenraten und Lebensdauer Einbruch)

Daher spezielle Anforderungen and OS und Filesystem

Nutzung von Bandlaufwerken

- manuell
- halbautomatisch (Volume-orientiert m. Operator)
- automatisch (Volume-orientiert m. Roboter)
- Direkt mit Applikation (heute äußerst selten!)
- Backup System
- Archiv System
- Hierarchical Storage Management (innovativ)
- Virtual Tape Library (innovativ)

Band Roboter

- Bestandteile:
 - 1 oder mehr Laufwerke
 - Speicherplätze für Band-Kassetten
 - 1 oder mehr Roboter-Arme für Kassetten
 - Erweiterungsschränke ohne Laufwerke
 - Kassetten haben eindeutige Kennzeichnung
- Bänder werden von Dienstprogramm verwaltet
- Roboter werden via SAN benutzt
- Anschluss an mehrere Rechner möglich

Backup System

- Daten werden manuell / automatisch gesichert
- Sicherungs-Typen
 - Voll-Backup (alle Daten)
 - Inkrementell (nur veränderte; versch. Level)
- Rückholung für End-User möglich
- Zusammenarbeit mit Applikationen (Datenbank)
 - Serverless Backup (Filesystem generiert Blockliste)
- Ggf. Verbindung mit Archiv System
- Wichtig: Restore muss eingeplant werden!

Archiv System

Archivierung von Daten \neq Backup!

- Auslagerung
 - Anderer Brandabschnitt
 - Tresor
 - Anderer Standort (Disaster Recovery)
- Bestimmte Stände zum Wiederabruf
- Einbeziehung von anderen Medien
 - WORM
 - Optical Tape
 - Micro-Fiche

Hierarchical Storage Management

- Verbindung von Platte und Band
 - Lange unbenutzte Daten wandern auf Band
 - Dateien bleiben transparent im Zugriff
 - Zugriff auf ausgelagerte Dateien langsamer (~1 Min)
 - Policies für Auslagerung weit definierbar
- Implementierung
 - Hooks im Filesystem
 - Filesystem für *staging*
 - Kombination mit Backup und Archiv

Virtual Tape Libraries

Virtualisierung des Band Roboters

- Plattenschrank
- Controller, der einen Band Roboter simuliert
- Häufig zusammen mit echtem Tape Roboter

Wirkt wie Cache für den Band Roboter

- Weniger (teure) Laufwerke notwendig
- Laufwerke werden immer optimal betrieben
- Schreiben auf viele Laufwerke möglich
- Mehr Performance bei geplantem Lesen

Boot Prozess

Vom ausgeschalteten Rechner ...

... zum laufenden OS mit Applikationen

- Prozessor hat eingebaute Power-ON Aktion
- Service-Prozessor
- Einstufig / Mehrstufig

Mainframe

Initial Program Load (via Service Prozessor)

- Kanal-Befehl zum Lesen in Hauptspeicher
 - Device einstellbar
 - Adresse einstellbar
- Start des gelesenen Blockes
 - in *kernel mode*
 - in *real memory mode*
- Dann Laden des Bootstrap Loaders
- Dann Laden des OS oder der HW-Virtualisierung (LPar, z/VM)

PC – BIOS

- Prozessor startet Programme aus PROM
 - Selbsttest
 - BIOS jeder Karte (PCI und andere)
- Prozessor lädt *boot record*
(1. Sektor vom eingestellten Gerät)
 - MBR (wenn vorhanden und gültig)
 - *boot record* der aktivem Partition
- Der *boot record* lädt den Bootstrap loader
 - Block I/O durch BIOS schon verfügbar
- Bootstrap lädt das Betriebssystem
 - Enthält Filesystem Code zum Lesen

PC – GRUB

Wie PC mit BIOS

- Bootstrap-Code enthält Blockliste vom Rest
- Filesystem Modul lädt Grub
 - Auswahl Menu
 - Ggf. Graphik für Hintergrund
- Grub Aktion:
 - Direkt OS in Partition booten
 - Andere Partition ab Bootstrap booten

OpenBoot PROM (SPARC, ...)

- OBP ist ein standalone Forth System
- Zugriff auf alle Devices
- OBP Code in PCI Karten (Norm!)
- Device Zugriffs-Strukturen und Module in Forth
- ggf. UFS Filesystem Code

Bootvorgang:

- OBP lädt Boot Sektor
- Boot Sektor lädt Bootstrap
- Bootstrap lädt OS Kernel
 - Direkt wenn Boot Sektor den UFS-Code im OBP erkennt