

LI250 Linux Performance Analysis & Tuning

Kurzbeschreibung:

Systemplaner und Administratoren mit fortgeschrittenen Linux-Kenntnissen lernen, Performance-Engpässe zu erkennen, zu analysieren und nachhaltig zu beheben. Vermittelt werden Kernel- und Hardwareaspekte, CPU- und Speicherverwaltung, Massenspeicher, Dateisysteme, Netzwerk-Performance sowie Benchmarking und Monitoring mit Tools wie collectd und RRD.

Zielgruppe:

Das Seminar **LI250 Linux Performance Analysis & Tuning** ist geeignet für:

- Systemplaner
- Systemadministratoren

Voraussetzungen:

Um den Kursinhalten und dem Lerntempo des Workshops **LI250 Linux Performance Analysis & Tuning** gut folgen zu können, sollten Sie über folgende Vorkenntnisse verfügen:

- Fortgeschrittene Linux-Kenntnisse
- I/O- und Netzwerk-Grundlagen
- Hardware-Grundkenntnisse
- Grundkenntnisse in einer Skriptsprache sind von Vorteil

Sonstiges:

Dauer: 5 Tage

Preis: 2790 Euro plus Mwst.

Ziele:

Der Kurs **LI250 Linux Performance Analysis & Tuning** vermittelt Kenntnisse, um im Betrieb Performance-Engpässe zu erkennen und einzuordnen, Performance-Tests durchzuführen und Linux-Systeme anhand deren Hard- und Software zu bewerten. Praxisorientierte Hinweise für eine optimale Dimensionierung neuer Systeme, performance-orientierten Aufbau von Anwendungs- und Dienste-Infrastrukturen sowie Tuning-Möglichkeiten auf System- Ebene ergänzen den Kurs.

Inhalte/Agenda:

- Die Performance von vorhandenen Linux-Systemen zu analysieren und letztlich zu steigern, sollte für alle Administratoren ein interessantes Thema sein. In dieser Schulung werden folgende Themen ausführlicher behandelt:
- - ◆ **Linux Kernel**
 - ◆ Überblick über performance-relevante Funktionen
 - ◆ Aktuelle performance-relevante Entwicklungen
 - ◆ Überblick /proc und /sys-Dateisystem
 - ◆ Kontrollgruppen (Control Groups)
- - ◆ **Hardware-Aspekte**
 - ◆ Architektur eines modernen Computersystems
 - ◆ Unterschiedliche Bus-Systeme
 - ◆ Verschiedene Storage-Systeme
 - ◆ Software- und Hardware-RAID
 - ◆ IOPS
 - ◆ Besonderheiten von Flashspeicher
- - ◆ **Prozessor**
 - ◆ Der Process Scheduler
 - ◆ CPU-gebundene Workloads erkennen
 - ◆ Metriken auf System- und Process-Level
 - ◆ Scheduling-Strategien und CPU-Affinität
 - ◆ Processes im Detail: States und Tracing
 - ◆ Frequenzskalierung
- - ◆ **Hauptspeicher**
 - ◆ Grundlagen zur Linux-Speicherverwaltung
 - ◆ Hauptspeicher-gebundene Workloads erkennen
 - ◆ Metriken auf System- und Process-Ebene
 - ◆ Auslagerungsverhalten und Out Of Memory Killer
 - ◆ Shared Memory
 - ◆ Limits für 32-Bit und 64-Bit Linux
 - ◆ Wieviel Speicher braucht ein Prozess?
 - ◆ Exkurs: Speicherleck erkennen
 - ◆ Exkurs: Speicherverbrauch begrenzen
 - ◆ Exkurs: Hauptspeicher komprimieren
- - ◆ **Massenspeicher und Dateisysteme**
 - ◆ Processes I/O generieren
 - ◆ Linux I/O Stack,
 - ◆ I/O-gebundene Workloads erkennen
 - ◆ Metriken auf System-, Dateisystem- und Prozess-Ebene
 - ◆ Empfohlene Dateisystem-Größen
 - ◆ I/O-Scheduling, Multipathing
 - ◆ Besonderheiten von SAN-Systemen und Flashspeicher
 - ◆ Unterschiedliche Dateisysteme wie ext2-4 und XFS
 - ◆ Ausblick: BTRFS
 - ◆ Performance und Datenintegrität
 - ◆ Performancerelevante Mkfs- und Mount-Optionen
 - ◆ Dateisysteme ausrichten
 - ◆ Handhabung von SSDs
- - ◆ **Netzwerk**
 - ◆ Sende- und Empfangspuffer
 - ◆ Congestion Control
 - ◆ Analyse von TCP-Verbindungen
 - ◆ Netfilter/Conntrack Parameter
 - ◆ Bufferbloat
 - ◆ Statistiken und Netflows
- - ◆

- ◆ **Methoden der Performance-Analyse**

- ◆
 - ◇ Einige Antimethoden
 - ◇ Systematische Methoden
 - ◇ Die USE-Methode

- ◆ **Graphing und Monitoring**

- ◆
 - ◇ Grundbegriffe
 - ◇ Real-Time Monitoring
 - ◇ RRD Grundlagen
 - ◇ Datensammeln mit collectd

- ◆ **Benchmarks**

- ◆
 - ◇ Korrekte Messung und typische Messfehler
 - ◇ Unterschiedliche Benchmark-Typen
 - ◇ Test-Szenarios mit gängigen Benchmarks
 - ◇ Belastungsgrenzen herausfinden
 - ◇ Ergebnisse bewerten
 - ◇ Exkurs: Flexible I/O Tester

- ◆ **Anwendungen**

- ◆
 - ◇ NFS und FS Cache
 - ◇ Apache
 - ◇ MySQL und PostgreSQL
 - ◇ Rsync
 - ◇ Ausblick: Load Balancing via IPVS