PRAKTIKA

Oracle Ankopplung an ein Prolog basiertes Multidatenbank System

Auf unserem Institut wurde eine effiziente Prolog–Oracle Schnittstelle entwickelt. Im Rahmen dieses Praktikums ist das Oracle Datenbanksystem unter Verwendung dieser Schnittstelle an ein ebenfalls in Prolog geschriebenens Multidatenbank System (MDBS), das Abfragen über mehrere, verteilte Datenbanksysteme erlaubt, anzubinden. Dabei ist es notwendig, die Prolog–Oracle Schnittstelle genau zu verstehen und dann geeignete Methoden zu entwickeln, wie der Datentransfer am effizientesten zu gestalten ist.
Publikationen über die Prolog–Oracle Schnittstelle und das Prolog basierte MDBS sind elektronisch verfügbar: <ftp://ftp.complang.tuwien.ac.at/pub/eva/home.html>.
Vorkenntnisse: Systemprogrammierung, Datenbanksysteme, Prolog
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Deduktives Datenbanksystem

Sequentielles Prolog soll als einfaches, deduktives Datenbansystem verwendet werden. Dazu gehören: (⋆) persistente Speicherung großer Datenmengen, (⋆) Transaktionen und Ausfallsicherheit der Daten, (⋆) Entwicklung einer Prolog basierten Datenmanipulations- und Datenabfragesprache, (⋆) Triggermechanismus und die (⋆) automatische Überprüfung von Konsistenzbedingungen.
Zu jeder der notwendigen Erweiterungen gibt es bereits Vorarbeiten und Publikationen, die weiterverwendet werden können. (Die Diplomarbeit "Graphische Benutzeroberfläche für Deduktive Datenbanken" hängt thematisch mit diesem Praktikum zusammen.)
Vorkenntnisse: Systemprogrammierung, Datenbanksysteme, Prolog, C
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Verteiltes Programmieren

Wir stellen Ihnen eine Reihe verschiedener neuer "high-level" Werkzeuge für verteiltes Programmieren auf verteilter und paralleler Hardware zur Verfügung, wie z.B. PVM (Parallel Virtual Machine), ISIS, C-LINDA und CoKe (Coordination Kernel). Sie lösen mit diesen Werkzeugen "klassische" verteilte Problemstellungen (z.B. "branch and bound", "alpha beta search") und gewinnen Erfahrung in der Programmierung verteilter Systeme. Focus ist die Gegenüberstellung der verschiedenen Werkzeuge und ihrer Philosophie. Genaue Benchmarks sollen in dieser Arbeit durchgeführt werden.
Vorkenntnisse: Programmiersprachen, Netzwerke
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Fehlertolerantes Programmieren im Netzwerk

Computer können abstürzen. Auf einem einzelnen Rechner bedeutet das das Ende aller Berechnungen. Auf Rechnern, die über Netzwerke verbunden sind, sollte der Ausfall eines Rechners nicht die gesamte verteilte Berechnung stillegen. Weiters ist zu überlegen, welche Daten Systemabstürze zu überleben haben.
Im Rahmen dieses Praktikums sind diese Anforderungen an Hand von ein paar klassischen verteilten Programmen zu untersuchen. Wir stellen Ihnen einen Werkzeugkasten zur Verfügung, der bereits Datenpersistenz garantiert und Ihnen die Möglichkeit gibt, Wiederaufsetzpunkte für Berechnungen zu spezifizieren und es somit leicht macht, robuste, verteilte Programme zu schreiben. Focus

ist der praktische Erfahrungsgewinn: was bedeutet "Recoverability" für die einzelnen Problemstellungen? Eines unserer DEMO-Beispiele ist etwa ein "recoverable" Primzahlenberechner, bei dem nach Abstürzen jeder Arbeiter wieder seinen alten Job weiterführt. Das Testen soll nicht nur durch SIGNAL schicken, X-Terminal-Ausschalten, sondern auch durch systematische "Fehlerinjektion" erfolgen.
Vorkenntnisse: Programmiersprachen, Netzwerke
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Graphische Benutzeroberfläche für ein Multidatenbank System

Ein Multidatenbank System (MDBS) erlaubt den Zugriff auf autonome Datenbanken, die im Netwerk verteilt sind. Unter einer "heterogenen Transaktion" versteht man eine Transaktion, die unter Wahrung der lokalen Autonomie sowie der lokalen und globalen Konsistenz, mehrere Datenbanksysteme involviert. So ein MDBS wurde von uns in Prolog realisiert. Im Rahmen dieses Praktikums ist eine Metapher zu entwickeln, WIE der ungeschulte MDBS Benutzer die verteilten Datenbestände sehen sollte. Wie kann die Komplexität, die durch die Verteilung entsteht, am besten visualisiert werden? Wie sieht ein geeigneter MDBS Browser aus? Was sind die Unterschiede zu einem herkömmlichen UFI ("User Friendly Interface") einer zentralisierten Datenbank?
Publikationen über das Prolog basierte MDBS sind elektronisch verfügbar:
<ftp://ftp.complang.tuwien.ac.at/pub/eva/home.html>.
Vorkenntnisse: Systemprogrammierung, UNIX, C, X-Windows
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Systematisches Testen eines neuen Software Protoytyps für verteilte Systeme

Wir haben einen Werkzeugkasten namens CoKe (Coordination Kernel) implementiert, der verteiltes und paralleles Programmieren einfach macht. Der CoKe kann in beliebige Gastsprachen eingebettet werden. Bereits fertig realisiert ist die Einbettung in Prolog, die Prolog&Co heißt ("Prolog plus Coordination"). Prolog&Co steht samt zugehöriger Dokumentation im Internet zur Verfügung.
Im Rahmen dieses Praktikums ist systematisch zu testen, ob die Prolog&Co Implementierung die in der Dokumentation vorgegebene "Requirement Specification" erfüllt. Diese Aufgabe entspricht dem "Abnahmetest" eines Produkts.
Publikationen über den CoKe und seine Anwendungen sind elektronisch verfügbar:
<ftp://ftp.complang.tuwien.ac.at/pub/eva/home.html>.
Vorkenntnisse: keine
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Ausfallsichere, effiziente Datenspeicherung

Um Daten gegen Systemausfälle zu schützen, müssen sie in regelmäßigen Zeitabständen auf ein sicheres Speichermedium geschrieben werden. Dabei ist zu garantieren, daß diese Daten mit großer Wahrscheinlichkeit Fehler überleben. In Datenbanksystemen werden beispielsweise zwei Files verwendet, um das atomare Schreiben vom Daten beim Transaktionsende zu garantieren: ein Daten und ein LOG-File, von denen zu jedem Zeitpunkt garantiert werden kann, daß zumindest eines in Ordnung ist. Wir verwenden für ein von uns entwickeltes System ähnliche Techniken.
Im Rahmen dieses Praktikums ist die Sicherheit des von uns verwendeten Mechanismus im Vergleich zu anderen "State Storing" Technologien im Bereich persistenter Systeme zu untersuchen (hinsichtlich Überlebenssicherheit und Effizienz). Focus ist daher ein genaues Literaturstudium. Vorschläge zur Verbesserung der Überlebenssicherheit sind auszuarbeiten (dafür können eventuell zusätzliche Hardware oder zusätzliche Replikation der Daten im Netzwerk verwendet werden).
Vorkenntnisse: Datenbanken, UNIX, Systemprogrammierung
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Zugriffskontrolle in verteilten Systemen

Verschiedene Systeme, die Mechanismen zur Zugriffskontrolle anbieten, wie etwa das am MIT entwickelte KERBEROS sind zu untersuchen. Focus ist daher ein genaues Literaturstudium. Die Verwendbarkeit im Rahmen eines von uns entwickelten, verteilten und fehlertoleranten Betriebssystemkonzepts ist zu analysieren. Die "process" Primitive unseres Systems, die das Aufspannen eines "remote threads" auf einem anderen Rechner bewirkt und somit System und Sicherheitsgrenzen überschreitet, ist um den ausgewählte Mechanismus zu erweitern; respektive kann auch ein eigener, besserer Vorschlag realisiert werden.
Vorkenntnisse: verteilte Systeme
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Simulation der "Distributed Data Driven Prolog Machine 3DPAM" im Netzwerk

Die 3DPAM wurde am KFKI in Budapest entwickelt und ist ein paralleles, datenflußgesteuertes Berechnungsmodell für Prolog. Die 3DPAM wurde auf Transputer Architekturen realisiert. Die Aufgabe dieses Praktikums ist es, die 3DPAM in einem Netzwerk von verteilten Workstations zu simulieren. Dazu stellen wir Ihnen Prolog&Co, eine neue, Prolog basierte Sprache für verteiltes Programmieren zur Verfügung.
Vorkenntnisse: Prolog, verteilte Systeme
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Ein verteiltes Spiel fürs Internet

Um unsere Werkzeuge zur Programmierung verteilter Systeme in Einsatz zu bringen, soll mit ihrer Hilfe ein verteiltes Spiel im Netzwerk implementiert werden. Dabei ist zu beachten, daß die Spieler im regelmäßigen Abständen miteinander kommunizieren müssen (kooperativ und Kampf um gemeinsame Ressourcen) und daß der "scale-up" beliebig sein soll, d.h., beliebig viele Spieler sollen dynamisch hinzukommen respektive aufgeben können. Ein Netzwerk oder Systemfehler soll den Spielverlauf nicht kompromittieren.
Die Regeln und das Layout des Spiels können selbst gewählt oder erfunden werden. Eine einfache, graphische X-Oberfläche soll implementiert werden.
Vorkenntnisse: Netzwerke, UNIX, Programmiersprachen
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Typüberprüfung für Prozeß-Typen

Mehrere moderne Programmiersprachen unterstützen kommunizierende parallele Prozesse. Typen in einigen dieser Sprachen können die Dynamik des Sendens und Empfangens von Nachrichten beschreiben und sind daher mächtiger als Typen in herkömlichen sequentiellen Sprachen. Es soll ein Prolog-Programm geschrieben werden, das Programme in einer solchen Sprache auf Typkonsistenz prüft.
Vorkenntnisse: Prolog
Betreuung: Franz Puntigam (Tel: 58801/4458)

Konsistenzprüfung unvollständiger Spezifikationen

In einigen neuen objektorientierten Sprachen entsprechen Typen unvollständigen Spezifikationen der dazugehörigen Objekte. Durch Typ-Vererbung („subtyping") werden solche Typen zueinander in Beziehung gesetzt. Es soll ein Prolog-Programm geschrieben werden, das überprüft, ob je zwei Typen (Spezifikationen) zueinander in einer solchen Beziehung stehen.
Vorkenntnisse: Objektorientierte Programmierung, Prolog
Betreuung: Franz Puntigam (Tel: 58801/4458)

Implementierung eines Interpreters

Es soll ein Interpreter für eine neue objektorientierte und parallele Programmiersprache geschrieben werden.
Vorkenntnisse: Übersetzerbau, Objektorientierte Programmierung
Betreuung: Franz Puntigam (Tel: 58801/4458)

Globale Registerbelegung

Um Konflikte zwischen den Registerbelegungsphasen zu verhindern, soll der lokale Registerallokator des GNU C Compilers eliminiert werden. Der globale Registerallokator soll auf den von Briggs beschriebenen Algorithmus umgestellt werden.
Betreuung: Anton (Tel: 58801/4459)

Registerbelegung und Befehlsanordnung

Diese beiden Compilerphasen machen einander das Leben schwer: Diejenige, die zuerst kommt, behindert die andere. In der Literatur wurden mehrere Methoden vorgeschlagen, diese Phasen aufeinander abzustimmen. Eine dieser Methoden sind im Rahmen des GNU C Compilers zu implementieren und zu bewerten.
Betreuung: Anton (Tel: 58801/4459)

Übersetzung von Forth in Maschinensprache

Forth is eine interaktive Programmiersprache, deren bekanntestes Merkmal die offen verwendeten Stacks sind (ähnlich Postscript und HP-Taschenrechnern). Im Moment existiert ein einfacher Übersetzer von Forth in MIPS-Maschinensprache. Er soll erweitert werden um: bessere Befehlsauswahl, bessere Befehlsanordnung, Tail-Call-Optimierung, und Codeerzeugung für andere Architekturen (z.B. Intel).
Betreuung: Anton (Tel: 58801/4459)

Ein Garbage Collector für VIP-Prolog

Der Interpreter für VIP-Prolog soll um einen kompaktierenden Mark-Scan Garbage Collector erweitert werden.
Betreuung: Andi (Tel: 58801/4462)

ISO-Prolog-konforme eingebaute Prädikate für VIP-Prolog

Die I/O-Prädikate sind an die neue ISO-Norm anzupassen.
Betreuung: Andi (Tel: 58801/4462)

Prolog-Compiler

Der MIPS-Codegenerator soll durch einen SPARC-Codegenerator ersetzt werden.
Betreuung: Andi (Tel: 58801/4462)

Threaded Code für Prologinterpreter

Threaded Code ist die schnellste Implementationstechnik für Interpreter. Leider kann sie nicht in Standard-C implementiert werden, mit einer GNU-C-Erweiterung geht es aber. Ein in C geschriebener Prolog-Interpreter ist auf Threaded Code umzustellen.
Betreuung: Anton (Tel: 58801/4459)

Ein seiteneffektfreier Prologparser

Die meisten praktischen Implementierungen von Parsern in Prolog verwenden Seiteneffekte zum Lesen einer Datei. Durch neuere Implementierungstechniken ist es möglich geworden, da eine Datei direkt auf eine Liste von Zeichen effizient abgebildet werden kann. Dadurch erübrigt sich die Verwendung von Seiteneffekten innerhalb eines Parsers. Es soll ein Parser für Prolog entwickelt werden, der eine Liste von Zeichen auf den entsprechenden Prologterm abbildet. Dieser Parser soll mit existierenden Parsern verglichen werden.
Vorkenntnisse: Prologübung
Betreuung: Ulrich Neumerkel, ulrich@mips.complang.tuwien.ac.at

Linux Disk-Management fr Laptops

Zur Schonung der Batterie auf einem Laptop sollen die Plattenzugriffsstrategien von Linux adaptiert werden. Üblicherweise wird durch den update-Dämon immer wieder auf die Platte zugegriffen. Dadurch ist es nicht möglich, die Platte zur Einsparung der Batterie abzuschalten. (Eine einfache, aber nicht optimale und zufriedenstellende Lsung ist es, den update-Dämon erst gar nicht zu starten und manuell zu sync-en.) Es soll eine für Laptops geeignetere Strategie implementiert werden, die die Zeit zu der die Platte abgeschaltet sein kann, maximiert.
Vorkenntnisse: Linux-Kernel
Betreuung: Ulrich Neumerkel, ulrich@mips.complang.tuwien.ac.at

Linux Prozess-Memory-Management

Prozesse teilen gewisse Hauptspeicherbereiche miteinander. Etwa die nicht modifizierbaren Teile unabhängiger Prozessinstanzen, Bibliotheken und die nicht modifizierten Teile zwischen Vater- und Kindproze. Daten, die vom Proze selbst initialisiert wurden, werden jedoch nicht mit identischen Daten eines anderen Prozesses zusammengelegt. Durch das Zusammenlegen solcher Daten wird der Hauptspeicherbedarf und damit auch das Swapping (paging) in Linux weiter reduziert. Es soll a) gemessen werden, welche Speichereinsparungen dadurch erzielt werden knnten und b) der Linux-Kernel daraufhin adaptiert werden.
Vorkenntnisse: Linux-Kernel
Betreuung: Ulrich Neumerkel, ulrich@mips.complang.tuwien.ac.at

Ein Compiler fr binäres Prolog

Binäres Prolog ist eine stark vereinfachte Prologvariante, die verhältnismig einfach zu implementieren ist. Es soll ein Compiler entwickelt werden, der BinWAM-Code erzeugt.
Vorkenntnisse: Prologübung
Betreuung: Ulrich Neumerkel, ulrich@mips.complang.tuwien.ac.at

Schnittstelle für ein interaktives Programmtransformationssystem für Prolog

Es soll eine Schnittstelle entworfen werden, um Programmtransformationen in der Programmierumgebung GUPU interaktiv durchführen zu können.
Vorkenntnisse: Prologübung
Betreuung: Ulrich Neumerkel, ulrich@mips.complang.tuwien.ac.at

Ein Emacs-Mode für Prolog

Fr die Programmierumgebung GUPU wurde ein spezieller Emacs-Mode entwickelt, der das Entwickeln und Testen von Prologprogrammen unterstützt. Es soll ein allgemeiner einsetzbarer Mode zum Entwickeln und Testen von Prologprogrammen entwickelt werden.
Vorkenntnisse: Emacs und Emacs-Lisp
Betreuung: Ulrich Neumerkel, ulrich@mips.complang.tuwien.ac.at