DIPLOMARBEITEN

Ein Kommunikationsprotokoll für multimediale Daten

CoKe (Coordination Kernel) ist ein Werkzeugkasten, der verteiltes und paralleles Programmieren einfach macht. Der CoKe unterstützt den gemeinsamen Zugriff auf verteilte Datenobjekte. Auf jedem Rechner läuft ein CoKe, der mit den CoKes auf den anderen Rechnern mittels geeigneter "Kommunikationsprotokolle" redet und den Objektzugriff regelt (z.B. den exklusiven Schreibzugriff auf Objekte). Es gibt bereits zwei Protokolle, die auf Objektreplikation basieren.
Im Rahmen dieser Diplomarbeit ist ein weiteres Protokoll zu entwickeln, das für die Verwaltung sehr großer, multimedialer Datenmengen geeignet ist. Das Kopieren der Objekte auf jeden beteiligten Rechner muß durch geeignete Cachingmethoden umgangen werden.
Publikationen über den CoKe und existierende Kommunikationsprotokolle sind elektronisch verfügbar: <ftp://ftp.complang.tuwien.ac.at/pub/eva/home.html>.
Vorkenntnisse: Netzwerke, Systemprogrammierung, C
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Verteilter, kommunikativer "Time Planner"

Zum Management unserer (so knappen) Zeitressourcen verwenden wir heute ca. A5-große Time Planner und zur Kommunikation von Terminen elektronische Mail. Es soll im Rahmen dieser Diplomarbeit ein Modell entwickelt werden, das das gesamte "Time Planning" elektronisch unterstützt. Memos und Termine sowie über E-Mail empfangene Informationen und Daten sollen integrierbar werden. Ein paar Ideen: Eine Mail kann sofort erledigt werden, oder triggert eine Terminvereinbarung, oder triggert mehrere Folgeaktionen. Eine wichtige Nachricht, ein wichtiger Termin soll nicht übersehen werden können. Der private "Time Planner" soll im Netzwerk als Server agieren, sodaß Kunden automatisch Termine reservieren können. Durch geeignete Regeln kann sich der Besitzer vor unangenehmen Terminen schützen. Wie kann zuätzlich ein "off-line Management" des Time Planners unterstützt werden?
Die Implementierung ist mit Hilfe unseres neuen Werkzeugkastens für verteiltes Programmieren zu realisieren. Empfehlenswert ist dabei die Verwendung von Prolog als Basissprache, um regelbasiertes Wissen geeignet modellieren zu können.
Vorkenntnisse: Programmiersprachen, Netzwerke, (Prolog)
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Eine Analyse des Problems globaler Namensgebung in verteilten Systemen

Um auf Daten zugreifen zu können, muß man sie entweder eindeutig benennen können oder eine eindeutige Referenz auf sie besitzen. Schon in kleineren Umgebungen ist die Wahl eindeutiger Namen ein Problem, mehr noch in verteilten Systemen, wo verschiedene Benutzer autonom Namen vergeben können. Diese Problematik findet sich bei verteilten Betriebssystemen genauso wie bei verteilten Programmiersprachen.
Im Rahmen dieser Diplomarbeit soll die Literatur zu diesem Thema eingehend studiert werden und eine systematische Aufstellung von verteilten Systemen gegeben werden, ob und wie dort das "global name space problem" gelöst wurde. Neben Forschungsarbeiten sollen inbesondere die im kommerziellen Bereich häufig verwendeten Systeme (z.B. CORBA, OSF-DCE) dahingehend untersucht werden. Auch das von uns entwickelte Konzept zur Verwaltung verteilter Datenobjekte soll bei dieser Analyse berücksichtigt werden. Eine kritische Bewertung soll die Analyse abrunden.

Vorkenntnisse: keine
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

"Simple Off-Line Document Management"

Der Zugriff auf gemeinsame Daten erfordert Synchronisation. Solange alle Beteiligten "on-line" sind und sozusagen auf eine gemeinsame Kopie zugreifen, ist das Problem noch relativ einfach. Sobald aber Kunden "off-line" auf dieselben Daten zugreifen wollen, müssen neue Technologien entwickelt werden, die die Konsistenz der Daten gewährleisten ("Mobile Computing").
Im Rahmen dieser Diplomarbeit sollen an Hand von ein paar einfachen Dokumenttypen Strategien zum "off-line Management" entwickelt werden. Als Beispiel: die Verwaltung von Adress- und MEMO-Dateien auf einem Rechner im Netzwerk (der beispielsweise am Arbeitsplatz steht) und off-line auf einem Laptop, der aber zeitweise über eine Modemverbindung mit den Rechnern im Netzwerk kommunizieren kann. Eine primitive Strategie wäre hier, anzunehmen, daß bei dieser Art von Dokumenten nur Daten hinzukommen, daher ein uneingeschränktes gleichzeitiges Weiterarbeiten on-line und off-line zu erlauben und beim Zusammenführen der Daten einfach eine Übermenge zu bilden.
Es sollen ein paar Vorschläge für verschiedene, klassische Dokumenttypen entwickelt werden, die mit anderen Ansätzen (Literaturstudium!) zu vergleichen sind.
Vorkenntnisse: keine
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

A Simulation of Vienna Parallel Logic in Prolog

VPL (Vienna Parallel Logic) ist eine Erweiterung von Prolog um Nebenläufigkeit, Kommunikation und Transaktionen. VPL wird für paralleles und verteiltes Programmieren verwendet. Es gibt bereits mehrere Prototypsysteme von VPL. Im Rahmen dieser Diplomarbeit soll VPL in Prolog simuliert werden und auf Rechnern, die im Internet verteilt sind, realisiert werden. Dazu kann ein Werkzeugkasten verwendet werden, der von sequentiellem Prolog aus aufgerufen wird und der sichere Kommunikation über Datenobjekte, den "remote" Aufruf von Berechnungen sowie Transaktionen unterstützt.
Die wesentliche Aufgabe dieser Diplomarbeit ist daher eine geeignete Simulation von lokalen "threads" mit entsprechendem Prozeßmanagement in Prolog.
Vorkenntnisse: Prolog
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Realisierung einer Work Flow Applikation

Work Flow Management umfaßt die Beschreibung von Arbeitseinheiten, sowie die Spezifikation von Daten- und Kontrollflüssen zwischen ihnen. Das inkludiert auch die Spezifikation von Konsistenzbedingungen und Abhängigkeiten. Im Rahmen dieser Diplomarbeit soll ein geeignetes Work Flow Problem gefunden werden, das auch praktisch Relevanz hat. Dieses Problem soll genau analysiert und dann mit Hilfe von geeigneten Koordinationswerkzeugen, die wir Ihnen zur Verfügung stellen und die bereits Work Flow Management unterstützen, implementiert werden. Ein Beispiel wäre die Abläufe in einer Versicherungsanstalt zu modellieren.
Publikationen über Work Flow Management sind elektronisch verfügbar:
<ftp://ftp.complang.tuwien.ac.at/pub/eva/home.html>.
Vorkenntnisse: Programmiersprachen
Betreuung: Univ. Doz. Dr. eva Kühn, 58801-4475, eva@mips.complang.tuwien.ac.at

Sprachkonzepte und Wiederverwendung

Als Vorteil der objektorientierten Programmierung wird meist die Unterstützung der Wiederverwendung von Code und Spezifikationen genannt. In der Praxis treten dabei jedoch häufig Probleme

auf. Es sollen die Ursachen für diese Probleme untersucht werden. Dabei sollen Zusammenhänge zwischen den Problemen und verschiedenen Unterparadigmen des objektorientierten Paradigmas bzw. den Konzepten der entsprechenden Programmiersprachen hergestellt werden.
Vorkenntnisse: Erfahrungen mit objektorientierter Programmierung
Betreuung: Franz Puntigam (Tel: 58801/4458)

Objektorientiertheit und Parallelität

Die objektorientierte und die parallele Programmierung sind zwei vielversprechende Themen in der Informatik. Bei der Verbindung der beiden Gebiete stößt man jedoch noch auf ungelöste Probleme. Diese Probleme sollen anhand von Literatur herausgearbeitet und Lösungsansätze gegenübergestellt und bewertet werden.
Vorkenntnisse: Objektorientierte Programmierung, Parallelverarbeitung
Betreuung: Franz Puntigam (Tel: 58801/4458)

Werkzeuge zur Entwicklung verteilter Systeme

In einer ausführlichen Literatursuche sollen Beschreibungen aktueller Programmiersprachen, Bibliotheken und Werkzeuge (PVM, Linda, MPI, Parallax, ...) zur Entwicklung verteilter Systeme zusammengetragen werden. Die Diplomarbeit soll einen umfassenden Überblick über den aktuellen Stand der Technik auf diesem Gebiet geben und Vergleiche ermöglichen. Um eigene Erfahrungen in die Arbeit einfließen zu lassen, wird es notwendig sein, den Umgang mit den wichtigsten Werkzeugen zu erlernen und diese zur Entwicklung von Beispiels-Programmen einzusetzen.
Vorkenntnisse: Grundzüge der Parallelverarbeitung
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. Die Methoden sind im Rahmen des GNU C Compilers zu implementieren und miteinander zu vergleichen.
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, globale/interprozedurale Registerbelegung, Minimierung der Stackpointeränderungen, Prozedurinlining, Tail-Call-Optimierung, und Codeerzeugung für andere Architekturen (z.B. Intel).
Betreuung: Anton (Tel: 58801/4459)

Codeerzeugung für Stackmaschinen

Stackmaschinen haben einen Stack statt des Registersatzes konventioneller Prozessoren. Analog zur Registerbelegung bei Registermaschinen ergibt sich das Problem, den Stack möglischst gut

auszunützen und die Hauptspeicherzugriffe zu minimieren. Entsprechende Methoden sind zu entwickeln und zu implementieren (z.B. im Rahmen von `lcc`, einem einfachen C-Compiler).
Betreuung: Anton (Tel: 58801/4459)

Abschätzung des Programmverhaltens

Die Optimierungsphase von Compilern verwendet Daten über das Verzweigungsverhalten und die Cache-miss-rates bestimmter Befehle. Solche Daten können in früheren Programmläufen gemessen werden. Für den praktischen Einsatz wäre es wichtig, diese Daten schon aus dem Programmtext abschätzen zu können.
Betreuung: Anton (Tel: 58801/4459)

Branch Prediction

Die besten Hardware-Verfahren zur Vorhersage des Sprungverhaltens erreichen 97% Genauigkeit. Können diese Regelmäßigkeiten auch von Compiler-Verfahren genutzt werden?
Betreuung: Anton (Tel: 58801/4459)

Integration von Klauselindizierung und Coroutining

Klauselindizierung eliminiert frühzeitig Klauseln, die ohne Seiteneffekt fehlschlagen, und beschleunigt dadurch die Ausführung. Coroutining schiebt die Ausführung von Zielen auf, deren Argumente noch nicht genügend instanziiert sind; es erleichtert die Programmierung. Beide müssen beim Aufruf eines Zieles die Argumente inspizieren. Die Integration würde Doppelinspektionen vermeiden und die Ausführung beschleunigen.
Betreuung: Anton (Tel: 58801/4459) **Vorkenntnisse:** Logikprogrammierung

Constraint Logic Programming

CLP-Sprachen haben neben dem Prolog-Kern noch einen Constraint-Löser eingebaut, der es z.B. erlaubt, aus den Constraints `A<B, A in 3..10, B in 1..5` zu schließen daß `A` 3 oder 4 ist und `B` 4 oder 5. In der CLP-Sprache Aristo sind derzeit alle Constraints in C implementiert. Es ist ein Mechanismus zu implementieren, der es erlaubt, Constraints in Aristo selbst zu schreiben.
Betreuung: Anton (Tel: 58801/4459) **Vorkenntnisse:** Logikprogrammierung

Ein Garbage Collector für VIP-Prolog

Der Interpreter für VIP-Prolog soll um einen kompaktierenden Mark-Scan Garbage Collector erweitert werden. Weiters soll die Freigabe des Speichers der Prolog-Datenbasis ermöglicht werden. Die Implementierung erfolgt in der Programmiersprache C. Die Algorithmen sind schon vorhanden.
Betreuung: Andi (Tel: 58801/4462)

Optimierender Objektcode Transformer

Programme auf neuen RISC Prozessoren können durch eine Umordnung der Befehle beschleunigt werden. Bei diesem Projekt soll durch Profiling gewonnene Informationen dazu genützt werden Sprungbefehle zu optimieren (branch prediction, branch aligning). Vorarbeiten für dieses Projekt sind vorhanden.
Betreuung: Andi (Tel: 58801/4462)

Prolog-Compiler

Der MIPS-Codegenerator für den VAM1p Compiler soll durch einen SPARC-Codegenerator ersetzt werden. Vorarbeiten für dieses Projekt sind vorhanden.
Betreuung: Andi (Tel: 58801/4462)

Prolog-Compiler in Prolog

Es ist ein Compiler von Prolog in die abstrakte Maschine WAM zu schreiben. Im Compiler sollen nur „reine“ Features von Prolog verwendet werden.
Vorkenntnisse: Logikprogrammierung; **Betreuung:** Anton (Tel: 58801/4459)