PARASITIC COMPUTING
Pflichtenheft

Version 1.0

Jürg Reusser
Luzian Scherrer

11. Juni 2002

Zusammenfassung

Das Prinzip „Parasitic Computing" wurde erstmals im August 2001 öffentlich erwähnt [BFJB01], als es Wissenschaftlern der Notre Dame Universität im US-Bundesstaat Indiana gelang, fremde Rechenkapazität ohne Wissen und Einwilligung derer Besitzer zur Lösung mathematischer Probleme zu nutzen. Die Vorliegende Diplomarbeit der Hochschule für Technik und Architektur Bern setzt auf dieser Forschungsarbeit auf und entwickelt Möglichkeiten, das vorgestellte Prinzip zu einem vollständig programmierbaren, parasitären Rechenwerk zu erweitern, welches sämtliche klassischen Probleme der Informatik berechnen könnte.

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangslage und Umfeld

Ende des Jahres 2001 haben Wissenschaftler der Universität Notre Dame im US-Bundesstaat Indiana eine Methode demonstriert, mit der man ohne Wissen und Einwilligung der Anwender deren Rechnerkapazität im Internet nutzen kann [BFJB01]. Sie lösten ein mathematisches Problem mit Hilfe von Web-Servern in Nordamerika, Europa und Asien, ohne dass die Betreiber dieser Sites etwas davon merkten. Dazu nutzten sie TCP aus: Um die Integrität von Daten sicherzustellen, platziert der Sender eines Datenpakets eine Prüfsumme über die im Datenpaket enthaltenen Datenbits im Header. Auf der Empfängerseite wird diese Prüfsumme nachgerechnet und je nach Ergebnis wird das Datenpaket als korrekt akzeptiert, oder es wird verworfen. Durch geeignete Modifikation der Pakete und deren Header gelang es einfache Berechnungen durchzuführen.

Diese Forschungsarbeit wurde in diversen Artikeln öffentlich publiziert und besprochen [Fre02], sie ist weitgehend auch aus technischer Sicht detailliert dokumentiert [oND02].

1.2 Vorhaben

Die mittels dieser Methode durchführbaren Operationen reichen prinzipiell aus, um jedes auf einem klassischen Computer lösbare Problem parasitär, sprich mit fremder Rechenkapazität, zu berechnen. In dieser Arbeit soll nun ein Compiler oder ein Interpreter entwickelt werden, der eine einfache Sprache für parasitäre Programme implementiert. Das heisst, ein solches Programm würde durch den Compiler/Interpreter automatisch in geeignete Grundoperationen übersetzt, die dann mittels dem gezeigten Missbrauch der TCP-Prüfsumme ausgeführt werden können.

1.3 Ziele

Es soll ein virtuelles, auf den parasitär durchführbaren Grundoperationen aufbauendes Rechenwerk (ALU) entwickelt werden, welches eine Teilmenge der gängigen arithmetischen und logischen Operationen zur Verfügung stellt. Diese Teilmenge muss mindestens dem Umfang an Instruktionen entsprechen, welcher benötigt wird, um sämtliche klassischen Probleme der Informatik zu lösen.

Um mit dieser virtuellen ALU arbeiten zu können, soll eine entweder durch sie interpretierbare oder für sie kompilierbare Assemblersprache entwickelt werden, die zusätzlich zu allen Instruktionen der ALU auch die nötigen Kontrollflussmöglichkeiten bietet. Es sind dies Sequenz, Auswahl und Wiederholung.

Da die virtuelle ALU in ihren Grundoperationen auf Netzwerkprotokollen und Verbindungen beruht und von deren Korrektheit abhängt, sollen geeignete Mechanismen zur Fehlererkennung und Fehlerbehebung untersucht und gegebenenfalls implementiert werden.

Neben der Muss-Anforderung einer Assemblersprache zur Programmierung des Rechenwerkes besteht optional die Möglichkeit eine Hochsprache (evtl. auf bereits existierenden Grundlagen der HTA-BE [Boi01]) zu erstellen. Eine solche Hochsprache

soll, falls sie realisiert wird, mittels eines Compilers in die Assemblersprache übersetzbar sein. Dies jedoch ohne die Anforderung, Effizienz steigernde Übersetzungsalgorithmen anzuwenden.

Die zum Abschluss resultierende Software soll in einem iterativen Zyklus aus sich erweiternden Prototypen erarbeitet werden. Im Realisierungskonzept (siehe Abschnitt 3.2 Seite 8), welches als nächstfolgender Meilenstein im Projekt gilt, werden die geforderten Ziele der Prototypen und des Endproduktes im Detail definiert.

1.4 Abgrenzung

1.4.1 Parasitäre Methode

Als parasitäre Methode wird für diese Arbeit primär das bekannte Vorgehen (vgl. [BFJB01]) mittels der Prüfsummen der Internetprotokolle auf den Layern 3 (IP, ICMP) und 4 (UDP, TCP) verwendet. Die Auswahl innerhalb dieser Kategorie wird im Realisierungskonzept (siehe Abschnitt 3.2 Seite 8) getroffen.

Die Erforschung und Evaluation weiterer, potenziell parasitär nutzbarer Protokolle (wie etwa solcher kryptographischer Natur) ist nicht vorgesehen, kann aber optional einbezogen werden falls entsprechende Varianten in Aussicht stehen.

1.4.2 Rechenwerk und Speicher

Das Rechenwerk soll ausschliesslich arithmetische und logische Operationen sowie Kontrollflussfunktionalitäten wie beispielsweise einen Programm-Counter bieten, die Register und der weiter benötigte Primärspeicher hingegen sind lokal abgebildet und nicht parasitär (sprich verteilt) zu implementieren.

1.4.3 Portabilität

Der Programmcode soll portabel und auf Linux sowie gängigen UNIX Systemen kompilierbar sein. Nicht unterstützt werden andere, sich wesentlich unterscheidende Betriebssysteme wie Microsoft Windows oder Apple MacOS.

1.4.4 Effizienz

Es ist nicht Ziel dieser Arbeit, die parasitären Berechnungen dahingehend zu entwickeln, dass sie dank ihrer Verteilung Effizienzvorteile gegenüber einer lokalen ALU bieten. Die Arbeit soll in diesem Sinne als ,,Proof of Concept” dienen und nicht in einem in der Praxis einsetzbaren High-Performance Rechner resultieren.

1.4.5 Parallelität

Wir beschränken uns auf die sequenzielle Abarbeitung von Programmcode und verfolgen keine Parallelisierung auf Level des Rechenwerkes wie dies etwa bei Multipipe-Prozessoren angewendet wird.

2 Projektorganisation

Beide Team Mitglieder (siehe Abschnitt 2.1 Seite 5) sind gleichberechtigt und neue Verantwortungsbereiche werden im Dialog fortlaufend aufgeteilt. Die Verantwortung für einen Bereich ist nicht gleichbedeutend mit alleiniger Durchführung.

2.1 Projektteam

Name	*E-Mail*
Luzian Scherrer	ls@parasit.org
Juerg Reusser	jr@parasit.org

2.2 Betreuer

Name	*E-Mail*
Dr. Jacques Boillat	jacques.boillat@hta-be.bfh.ch
Dr. Jürgen Eckerle	juergen.eckerle@hta-be.bfh.ch
Michael Dürig	michael.duerig@hta-be.bfh.ch

2.3 Experten

Name	*E-Mail*
Walter Eich	we@zuehlke.com

2.4 Arbeitszeiten des Projektteams

2.4.1 8. Semester, Projektphase

Während des 8. Semesters ist das in die Arbeit investierte Wochenpensum folgendermassen aufgeteilt:

Dauer	*Wochentag*	*Zeit*
6 Lektionen	Montag	16:15 bis 21:30
4 Lektionen	Dienstag	18:10 bis 21:30
6 Lektionen	Freitag	12:45 bis 17:50

Die Wochenarbeitszeit beträgt somit 16 Lektionen zu 45 Minuten, was 12 Stunden entspricht. Der Dienstag Abend wird für Besprechungen mit den Betreuern reserviert.

2.4.2 Diplomarbeitsphase

In der die Arbeit abschliessenden Diplomphase wird das Leistungspensum je nach Anforderungen und Stand des Projektes erhöht, wobei ein durchschnittlicher Aufwand von ungefähr 20 Wochenstunden anzustreben ist.

Besprechungen werden nach Rücksprache mit dem Experten respektive den Betreuern abgehalten.

2.5 Sitzungen

2.5.1 Teamsitzungen

Zweck	Projektstand, Terminüberwachung, Entscheidungen, aktuelle Probleme und weiteres Vorgehen
Teilnehmer	Projektteam
Häufigkeit	In der Regel jeden zweiten Dienstag Abend, 20:00 Uhr
Dauer	Nach Bedarf

2.5.2 Reviews

Zweck	Projektstand
Teilnehmer	Projektteam, Betreuer
Häufigkeit	Zwischen den Meilenstein-Reviews alle zwei bis drei Wochen
Dauer	Nach Bedarf

2.5.3 Meilenstein Review

Zweck	Meilenstein Abnahme
Teilnehmer	Projektteam, Betreuer, Experte
Häufigkeit	Beim erreichen eines Meilensteins
Dauer	Nach Bedarf

3 Projekspezifisches Vorgehensmodell

3.1 Projektstruktur

Die folgenden Abschnitte gliedern die Arbeit in ihre logischen Projektphasen ein.

3.1.1 Projekt-Setup

Das Projekt-Setup beinhaltet folgende Aktivitäten:

- Formelle Details klären
- Einarbeitung in die Materie
- Bereitstellung der Infrastruktur
- Konkretisierung der Problemstellung
- Definition der Ziele

Aus dieser Phase resultiert das fertige Pflichtenheft.

3.1.2 Erarbeitung Realisierungskonzept

Prinzipiell besteht die Implementation aus drei fundamentalen, erstrebenswerterweise möglichst lose gekoppelten, Schichten. Diese drei Teilbereiche sollen mittels vor der Implementierungsphase definierter Programmierschnittstellen verbunden werden können.

- Assembler
  - Literaturstudium
  - Existierende Sprachen auf Adaptierbarkeit und Umfang evaluieren
  - Benötigten Instruktionssatz festlegen
- Aufbau CPU/ALU
  - Literaturstudium
  - Studium und Festlegung der Architektur
  - Benötigte Datenstrukturen definieren
- Verteiltes Rechnen
  - Literaturstudium
  - Studium parasitär nutzbarer Netzwerkprotokolle
  - Potenzielle Fehlerquellen im Netzwerkteil eruieren
  - Untersuchung geeigneter Fehlerbehandlungsmechanismen

Aus dieser Phase resultiert das die Projektziele detailliert beschreibende Realisierungskonzept mit sämtlichen Angaben über die bevorstehende Implementation.

3.1.3 Implementation

In dieser Phase wird die effektive, auf dem Realisierungskonzept basierende Implementation durchgeführt. Dabei wird pro Modul sowie gesamthaft ein Vorgehen nach folgenden Punkten angestrebt:

- Analyse und Design
- Aufbau
- Durchführung
- Bewertung

3.1.4 Auswertung

Die Auswertungsphase umfasst folgende Teilbereiche:

- Gesamtbewertung
- Erreichte Ziele
- Erfahrungen
- Schlussbemerkungen

Aus der Auswertungsphase resultiert der abschliessende Projektbericht. Weitere Details zur Bewertung sind in Kapitel 5 festgehalten.

3.2 Meilensteine

Meilenstein	*Datum*
Abnahme Pflichtenheft	Mitte Juni 2002
Abnahme Realisierungskonzept	Vor den Herbstferien
Abschluss Implementation	Zwei Wochen vor Projektabnahme
Abnahme Projektbericht	Kalenderwoche 2, Januar 2003

3.3 Risikoanalyse: Risiken

3.3.1 Personenausfall (Krankheit, Militär, Beruf)

Faktor	Mittel
Auswirkung	Gross
Massnahmen	Permanentes Know-How-Sharing im Projektteam

3.3.2 Mathematisches Neuland

Faktor	Klein
Auswirkung	Gross
Massnahmen	Zusammenarbeit mit Mathematikern, entsprechende Know-How Beschaffung

3.3.3 Technisches Neuland

Faktor	Mittel
Auswirkung	Mittel
Massnahmen	Feingliedrige Bildung der Module, entwickeln von Simulatoren, falls Performanceprobleme

Im Realisierungskonzept werden Erfolgsaussichten und Risiken für die einzelnen Module/Komponenten separat abgeschätzt.

4 Ergebnisse

4.1 Besonderheit dieses Projekts

Bei dieser Diplomarbeit liegt das Schwergewicht nicht primär im Software Design sondern mehr im konzeptionellen und experimentellen Bereich. Dies hat natürlich auch Auswirkungen auf die zu erstellende Dokumentation. Insbesondere bedeutet dies, dass wir nicht eine Dokumentation im Sinne eines klassischen Softwareprojektes erstellen werden. Stattdessen werden Dokumente mit der Beschreibung der von uns durchgeführten Testimplementationen, Resultate und der gemachten Erfahrungen entstehen.

4.2 Dokumente

Bezeichnung	*Inhalt*
Pflichtenheft	Beschreibung der Ziele, Projektorganisation (gilt als definitive Aufgabenstellung)
Realisierungskonzept	Siehe Abschnitt 3.1.2 Seite 6
Implementation und Testbericht	Vorhaben Testen, Experimentieren, Resultate, Beurteilung
Lab Journal	Chronologische Notizen zum Projektverlauf, den Besprechungen und sonstigen Aktionen
Projektbericht	Projektverlauf, Erreichte Ziele, Bemerkungen, Schlusswort

4.3 Source-Code

Wir streben danach, die Applikation funktional betrachtet in verschiedene Module zu unterteilen, welche durch experimentieren und erforschen von Gegebenheiten und Ansätzen zur Lösungsfindung entwickelt werden. Wir behalten uns die Möglichkeit vor – falls dies als dringlich notwendig erscheint – vor Schluss der Diplomarbeit ein Redesign der ganzen Applikation vorzunehmen, da zum Zeitpunkt des Realisierungskonzeptes nicht hundertprozentig abgesehen werden kann, in welche exakten Richtungen sich verschiedene Lösungsansätze bewegen werden.

5 Beurteilung und Bewertung

Da diese Diplomarbeit nicht einem Projekt im Sinne der klassischen Software Entwicklung entspricht, bedarf es Korrekturen betreffend der Gewichtung der einzelnen Kriterien, welche in nachfolgender Bewertungsliste aufgeführt sind.

5.1 Bewertungsliste

	Arbeitsschritt	*Gewicht*
Vorbereitungsphase	Aufbau und Vollständigkeit des Pflichtenheftes	2
Durchführung	Arbeits und Zeitplanung	1
	Kreativität (Initiative, Selbstständigkeit)	3
	Wahl und Anwendung der (Arbeits-)Methodik	2
	Implementation, Robustheit, Programmierstil	2
	Systemtest (Verfahren, Durchführung, Bericht)	2
	Kommunikation mit Experten und Betreuer	1
Ergebnis	Übereinstimmung Endprodukt/Pflichtenheft bzw. Realisierungskonzept	5
	Allgemeiner Eindruck aus der Besichtigung	1
Projektbericht	Inhalt korrekt, vollständig, verständlich	3
	Sprache, Stil, Übersichtlichkeit	1
	Klare, aussagekräftige Zusammenfassung	1

6 Konfigurationsmanagment

6.1 Projektablage und Publizierung

Sämtliche Ergebnisse des Projektes werden laufend auf der Website `http://www.parasit.org` in ihrer jeweils aktuellsten Version verfügbar gemacht. Dokumente hierbei in den Formaten HTML und PDF, Source-Code und Binärdateien in TAR Archiven.

Über entscheidende Neuerungen wird mittels der Mailingliste `all@parasit.org` informiert. Dieser Verteiler beinhaltet das Projektteam und die Betreuer, optional auch den Experten.

Der Server wird voraussichtlich mit all seinen Funktionalitäten auch über Diplomschluss hinaus vollumfänglich zur Verfügung stehen.

6.2 Änderungskontrolle

Sämtliche Dateien tragen während der Bearbeitungsphase CVS interne Versionsnummern. Jeweils bei Abschluss einer Phase (Minor Release) sowie bei Abnahme (Major Release) wird eine symbolische Versionsnummer generiert. Minor Releases erhöhen hierbei die symbolische Versionsnummer in der Nachkommastelle, Major Releases in der Vorkommastelle. Die Änderungen zwischen Releases werden in Changelogs (in die Dokumente bzw. den Source-Code integriert) festgehalten und mitpubliziert.

6.3 Datensicherung

Sämtliche elektronisch gespeicherten Daten werden ausschliesslich mit dem Concurrent Versioning System [CVS02] verwaltet, womit jederzeit auf jede jemals existiert habende Version zugegriffen werden kann. Die persistente Datensicherheit ist durch das nächtliche Sichern des kompletten CVS Repositories in ein gewartetes Tape-Archive gewährleistet.

Literatur

[BFJB01] Albert-László Barabási, Vincent W. Freeh, Hawoong Jeong, and Jay B. Brockman. Parasitic computing. *Nature*, 412:894–897, August 2001.

[Boi01] Dr. Jacques Boillat. Code generierung. *Script*, 2001.

[CVS02] CVS. Concurrent versioning system. `http://www.cvshome.org/`, Juni 2002.

[Fre02] Vincent W. Freeh. Anatomy of a parasitic computer. *Dr. Dobb's Journal*, 332:63–67, Januar 2002.

[oND02] University of Notre Dame. Parasitic computing website. `http://www.nd.edu/~parasite/`, Juni 2002.