FHZ → FACHHOCHSCHULE ZENTRALSCHWEIZ

HTA → HOCHSCHULE FÜR TECHNIK+ARCHITEKTUR LUZERN

Abteilung Informatik

Diplomarbeit

Java

Thread Skalierung

Projektübersicht

HTA LUZERN
Technikumstrasse 21
CH—6048 Horw

T: 041-349-33-11
F: 041-349-39-60
W: www.hta.fhz.ch

Diplomarbeit 2006
Aregger Marcel
Meier Rainer

Änderungskontrolle

Version	Datum	Ausführende Stelle	Bemerkungen/Art der Änderung
1.1	2006-10-31	Marcel Aregger	Initial Release
1.2	2006-11-20	Rainer Meier	Review

Prüfung und Freigabe

Vorname/Name	Dokumentversion	Status	Datum	Visum
Rainer Meier	1.2	Final	2006-11-20	
Marcel Aregger	1.2	Final	2006-11-20	

1. Management Summary

1.1. Aufgabestellung

Physikalische Grenzen und fertigungstechnische Probleme bei der Erhöhung der Rechenleistung von Prozessoren verstärken zunehmend den Trend zu Multi-Core und Multi-CPU Systemen. Die Skalierung einer Applikation hängt nicht mehr primär von der Geschwindigkeit eines einzelnen Prozessors ab sondern vom Grad der Ausnutzung verfügbarer Recheneinheiten. Faktisch gesehen geht die Verantwortung „schnelle“ Applikationen zu realisieren vom Hersteller der Microchips auf die Entwickler der Anwendung über. Die Diplomarbeit „Java Thread Skalierung“ thematisiert die Einflussfaktoren der vertikalen Skalierung im Java-Umfeld.

1.2. Lösungskonzept

Die Bearbeitung dieser Problemstellung wurde gestützt auf einer phasenorientierter Planung in verschiedene Projektschwerpunkte gegliedert, die entsprechende Resultate und Dokumente hervorbrachten.

Im Bereich des Projektmanagements (SPMP) wurden aus der Aufgabestellung der HTA 20 Technische und 7 Methodische Zielsetzungen abgeleitet. Die Projektzielsetzungen {Z?} wurden sowohl vom Projektbetreuer als auch vom Experten freigegeben und bildeten die allgemeine Guideline für nachfolgende Projektphasen. Die Komplexität der Problemstellung erforderte die punktuelle Festlegung weitere Ziele. Sie können mit folgender Zielhierarchie dargestellt werden:

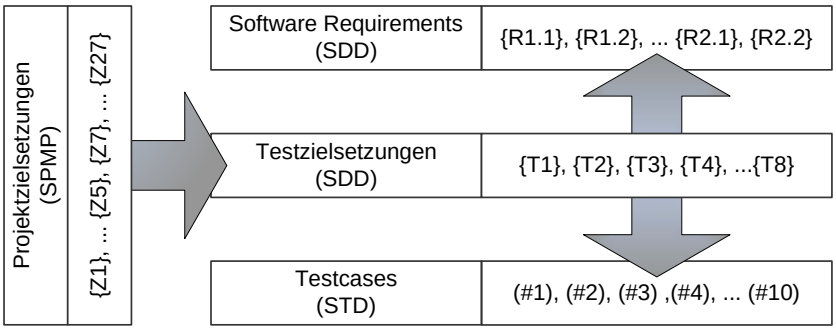

Abbildung 1 Zielhierarchie

Aus den Projektzielsetzungen {Z?} wurden im SDD Testzielsetzungen ({T?} = Testumfang) abgeleitet die wiederum die Basis für Software-Requirements ({R?}) und Testcases (#?) bildeten. Die Risikoanalyse im SPMP brachte 7 Hauptrisiken hervor, von denen das Risiko „Interpretation der Testresultate“ als hoch risikoreich und die Risiken „Unbefriedigender Projetoutput“ sowie „Phasengerechte Dokumentation“ als riskoreich eigestuft werden konnten.

Die Basisanalyse legte das Grundlagenwissen für die Arbeit im Bereich der Skalierung. In einer layerorientierten Betrachtung wurden für die Bereich Hardware, Betriebssystem, Applikation und JVM Einflussfaktoren gesucht, mit denen die Skalierung einer Applikation auf einem Computersystem beeinfluss werden können. Aus dieser Analyse konnte ein Sammelsurium an direkten und indirekten Faktoren extrahiert werden. Indirekt bedeutet in diesem Zusammenhang, dass diese Faktoren nicht oder nur mit Einschränkungen im Rahmen der Diplomarbeit umgesetzt werden können. Für den bereich Betriebssystem ergaben sich beispielsweise folgende Faktoren:

Tabelle 1 Einflussfaktoren Betriebssystem

Einfluss	Aspekt/Technologie
Direkt	Designprinzip Threads, WIN-32 Thread
Indirekt	Scheduling, Affinität

In der Phase der Umsetzung/Realisierung wurde im SDD die Implementierungs- und Testphase vorbereitet. Mit der Festlegung der Testplattform bestehend aus Technologien, Konzepten und Standards wurde festgelegt auf welcher „Plattform“ die Skalierung einer Java-Applikation untersucht werden sollte. Diese Abgrenzung berücksichtigte Aufgabenstellung, Zeit, Aussagekraft, Verfügbarkeit gleichermassen und führte zu folgender Testplattform:

Tabelle 2 Abgrenzung der Testplattform

Hardware	Betriebssystem	Applikation	JVM
- SMP	- Designprinzip Thread, - Win32 Thread, - Scheduling (Prioritäten), - Affinität	- OpenMP (in Form von JOMP)	- Java Threading, JOMP

Die Definition des geplante Testumfanges auf dieser Plattform mündete in eine Spezifikation für eine geeignete Testklasse. Mit der Berechnung der Mandelbrot-Menge wurde ein Algorithmus gefunden auf dem verschiedene Konzepte wie beispielsweise Methoden-/Objekt-Synchronisation, CAS (Lockfreie Synchronisation) oder JOMP (OpenMP) angewendet werden konnten.

Die effektive Umsetzung des geplanten Testumfanges (STD) erfolgte mit der Definition von 10 spezifischen Testcases aus den Bereichen Hardware, Betriebssystem und JVM. Diese Testreihe konnte mit einer 2-CPU-Architektur (SMP) und vergleichbarer 1-CPU-Architektur (Refernz) durchgeführt werden. Die Messresultate wurden systematisch erfasst und lieferten ausreichend Material für die Aanalyse.

Die Conclusion bildet mit der Interpretation der Messresultate den Abschluss der Testreihe. Die Schlussfolgerungen und daraus abgeleitete Implementierungs-Empfehlungen wurden dabei in die Bereiche Priorität, Affinität und Skalierung gegliedert. Mögliche Einsatzgebiete paralleler Anwendungen und ein kritisches Statement zum Einsatz von Multi-CPU-Systeme runden diese Phase ab.

1.3. Ergebnisse

Priorität

Im Bereich der Thread-Prioritäten konnte eine lineare Abbildung von Java-Thread- auf Kernel-Thread-Prioritäten nachgewiesen werden. Da jeweils 2 Java-Prioritäten auf eine Kernel-Priorität abgebildet werden, resultieren pro Process Priority Class faktisch 5 nutzbare Prioritätsstufen in Java. Wird das Priority-Mapping auf die Win32 Process Priority Classes ausgedehnt wird offensichtlich, dass im Bereich der Basisprioritäten Überlappungen entstehen die berücksichtigt werden müssen.

Affinität

Für eine multithreaded Java-Anwendung auf Windows XP resultiert eine gleichförmige Verteilung der Threads auf die verfügbaren CPUs des Systems sofern keine andere „Last“ um CPU-Zeit konkurriert. Über die Windows API können Affinitäten auf Level Thread und Prozess definiert werden. Mit Systemtools wie beispielsweise dem ProcessExplorer (Sysinternals) kann diese Zuweisung aber nur auf Prozess-Ebene definiert werden. Die Affinität wird in diesem Fall auf die Threads dieses Prozesses weitervererbt. Versuche unter Einwirkung von konkurrierenden Prozessen haben gezeigt, dass mit dem Setzen einer Affinität der jeweilige Prozessor nicht exklusiv zugeteilt wird.

Skalierung

Die Versuchsreihe bestätigte die 1:1-Abbildung eines Java-Threads auf einen Win32-Thread im Kernel. Somit laufen unter Windows XP die Threads einer Java-Anwendung als Kernel-Threads in einem Prozess-Kontext wo sie durch den Scheduler auf verfügbare CPUs verteilt werden können.

Eine multithreaded Java-Anwendung mit variabler Thread-Anzahl ohne Synchronisation dieser Threads zeigt zwischen der Single- und Multi-Prozessor-Architektur charakteristische Unterschiede. Während die 1 CPU-Architektur annähernd konstante Berechnungs- und CPU-Zeiten aufweist, werden auf der 2 CPU-Architektur die Berechnungszeiten bei zwei oder mehr Threads halbiert. Durch die effektive Verteilung der Threads auf 2 verfügbare CPUs wird Skalierungsfaktor von nahezu 2 erreicht.

Gemeinsam haben die Single- und Multi-Prozessor-Architektur, dass der zunehmende Verwaltungsaufwand (bis 512 Threads) zu keiner nennenswerten Zunahme der CPU-Zeit führt.

Für Anwendung in denen Synchronisation zum Einsatz kommt spielt auf einem Single-CPU-System das verwendete Verfahren eine entscheidende Rolle. Mit der Methoden-Synchronisation bricht beispielsweise mit mehr als 2 Threads die Skalierung um bis zu 50% ein. Wird mit Objekt- oder CAS synchronisiert ist es möglich eine annähernd konstante CPU- und Berechnungszeit wiederherzustellen.

Im 2 CPU-Umfeld ist die Auswahl des Synchronisationsverfahren nicht so zentral. Der charakteristische Verlauf ist bei allen Verfahren vergleichbar. Er zeigt für 2 Threads ein Skalierungsfaktor von annähernd 2 und einen mehr oder weniger konstanten Verlauf für eine höhrere Anzahl Threads in Bezug auf die CPU- und Berechnungszeit. Ein 2 CPU-System wird also mit der Verwendung einer Methoden-Synchronisation weit weniger ausgebremst als ein 1 CPU-System.

Erweiterte Testreihe

In Rahmen einer erweiterten Testreihe wurden spezifische Tests im Bereich Affinität und Skalierung auf einem MacBook Pro mit Intel Core 2 Duo Prozessor (C2D) durchgeführt. Diese Plattform ermöglichte die Analyse bzw. den Vergleich zwischen Windows XP und Mac OS X auf C2D. Alle Testresultate wurden im entsprechenden STD erfasst. Da diese Testreihe „out of project-scope“ durchgeführt wurde, konnten deren Resultate nur im eingeschränkten Rahmen interpretiert bzw. kommentiert werden.

1.4. Ablauf der Arbeit

Dank detaillierter Planung in der Startphase des Projektes konnten alle Arbeitspakete wie geplant abgearbeitet und die Termine gehalten werden. Die festgelegten Projektphasen mit Meilensteinen und erforderlichen Übergangskriterien (Teilresultate) erwiesen sich als praktikabel und realistisch.

Das Controlling des Projektes in Bezug auf den Arbeitsaufwand und die zu erarbeitenden Resultate wurde durch die Diplomanden selbst und den betreuenden Dozenten sichergestellt. Die Selbstkontrolle über die Projektplanung/Meilensteine und den Einfluss des Dozenten über die zahlreichen Reviews führten dazu, dass der Projektverlauf immer dem Planverlauf entsprach.

Ein Vergleich der Plan- und Ist-Werte ergibt einen Überblick über zeitliche, qualitative und quantitative Zielereichung in diesem Projekt:

Tabelle 3 Zielerreichung

Bereich	PLAN-Wert	IST-Wert
Aufwand	420 Std	441 Std
Dauer	6 Wochen	5.5 Wochen
Zielerreichung MUSS	22 (100% erfüllt)	21 (100%); 1 (nicht 100% erfüllt)
Zielerreichung KANN	5 (100% erfüllt)	2 (100%); 3(nicht 100% erfüllt)
Risiken	7 Hauptrisiken	5 Hauptrisiken eliminiert

Eine detaillierte Übersicht über den Grad der Zielerreichung bietet das Kapitel 6;Zielerreichung.

1.5. Dokumentation

Die Gesamtdokumentation dieser Projektarbeit umfasst in Papierform die Teildokumentationen gemäss Liste unter 7; Inhaltsübersicht Dokumentation. Für weiterführende Informationen bietet die Projekthomepage unter http://jts.skybeam.ch/ Hintergrundinformationen, JavaDoc, einen Downloadbereich der Teildokumente oder die Mandelbrot-Anwendung zum testen.

2. Inhaltsverzeichnis

3. Inhaltsübersicht Dokumentation

Das ausgedruckte Exemplar der Diplomarbeit enthält folgende Teildokumente:

Tabelle 4 Inhaltsübersicht Dokumentation

#	Dokumenttitel	Vers.	Beschreibung
1	Projektübersicht	1.2	Übersicht Diplomarbeit
2	Aufgabenstellung_DA06	1.0	Aufgabenstellung HTA
3	SPMP	1.9	Software Project Management Plan
4	SCMP	1.2	Software Configuration Management Plan
5	Basisanalyse	1.4	Grundlagen der Skalierung
6	SDD	1.4	Softeware Design Document
7	STD	1.3	Software Test Document
8	STD-MacBookPro	1.1	Software Test Document MacBook Pro
9	Testprotokolle	1.1	Testprotokolle STDs
10	Conclusion	1.2	Schlussfolgerungen Projektergebnisse
11	Glossar	1.1	Begriffserklärungen
12	JavaThreadSkalierung _Projektstatusbericht		Projektstatusbericht vom 20.10.2006 bis 27.10.2006
13	JavaThreadSkalierung _Projektstatusbericht 2.doc		Projektstatusbericht von 27.10.2006 bis 10.11.2006
14	03_Protokoll-Besprechung_20Okt06		Protokoll zur Besprechung vom 20. Oktober 2006
15	05_Reviewprotokoll – Grundlagen_Überprüfung		Protokoll zur Grundlagenüberprüfung vom 30.Oktober 2006
16	Reviewprotokoll Entwicklungsüberprüfung.pdf		Protokoll zur Entwicklungsüberprüfung vom 13 November 2006

4. Inhaltsübersicht CD

Die Verzeichnisstruktur auf der Projekt CD-ROM ist folgendermassen aufgebaut:

Tabelle 5 Verzeichnisstruktur

Verzeichnis	Inhalt	Beschreibung
Allgemein/	Aufgabenstellung_DA06.pdf DA_Ibb_2006_Zuteilung.pdf Leitfaden-Doku.pdf [Review Protokolle]	Enthält allgemeine Dokumentationen sowie auch die Review-Protokolle.
Analyse/	Basisanalyse.pdf	Basisanalyse
Conclusion/	Conclusion.pdf	Conclusion
Glossar/	Glossar.pdf	Globales Glossar
Poster/	Poster.pdf	Diplomposter
Präsentation/	Vorab-Praesentation.pdf Vorab-Praesentation.pps	Vorab-Präsentation im PDF und PowerPoint Format
Projektfortschritt/	JavaThreadSkalierung_Projektstatusbericht.pdf JavaThreadSkalierung_Projektstatusbericht 2.pdf Projektplanung-Java-Thread-Skalierung.mpp	Projektfortschritt und Statusberichte.
Projektübersicht/	Projektuebersicht.pdf	Projektübersicht (dieses Dokument)
Quellen/	[Diverse Quellen]	Archiv der Internet-Quellen
SCMP/	SCMP.pdf	Software Configuration Management Plan
SDD/	SDD.pdf	Software Design Document
Software/	JTSFraktal-dist/ JTSFraktal-JavaDoc/ JTSFraktal-src/ JavaDev_JumpStart_2006-11-16-win32.exe	Enthält den Quelltext, die Binaries sowie die JavaDoc der entwickelten Applikation. JavaDev_JumpStart enthält die Entwicklungsumgebung..
SPMP	SPMP.pdf	Software Project Management Plan
STD	STD.pdf, STD-MacBookPro.pdf System-Information.nfo Testprotokolle.pdf Testprotokolle.xls Benchmark-Place.xml	Software Test Document, Microsoft System Information 7 Dokumente, Testprotokolle, Mandelbrot Bildausschnitt
WebAbstract	[WebAbstract Dateien]	HTA WebAbstract
Webseite	[Webseite Dateien]	Inhalt von http://jts.skybeam.ch/

Hinweise:

- Das Verzeichnis `Software` enthält eine selbst installierende Java-Umgebung mit den folgenden Komponenten:
    - Apache ANT
    - Eclipse (inkl. Diverser Plugins)
    - Sun Java VM Version 1.5.0_09-b03
- `Software/JTSFraktal-dist/` enthält die direkt startbaren JAR-Dateien. Um sie ausführen zu können wird die Sun JavaVM 1.5.0 benötigt. Der Start von CD-ROM ist zwar möglich aber nicht empfohlen, da beim Beenden der Anwendung die Einstellungen und die Bookmarks nicht gespeichert werden können.
- `Software/ JTSFraktal-src/` enthält die Quelltexte sowie ein ANT Build-Skript um diese zu kompilieren von mandelbrot.jar sowie dem optionalen JOMP Plugin. Das ANT-Skript kann auch die ebenfalls enthaltene JavaDoc erstellen.

5. Risikoüberprüfung

Die Risikolandschaft zeigte zu Beginn des Projektes 7 Hauptrisiken, wovon 3 Risiken im Quadranten 1 und 2 positioniert wurden (Dringlichkeit mittel bis hoch). Die bereinigte Situation vor Projektabschluss präsentiert sich wie folgt:

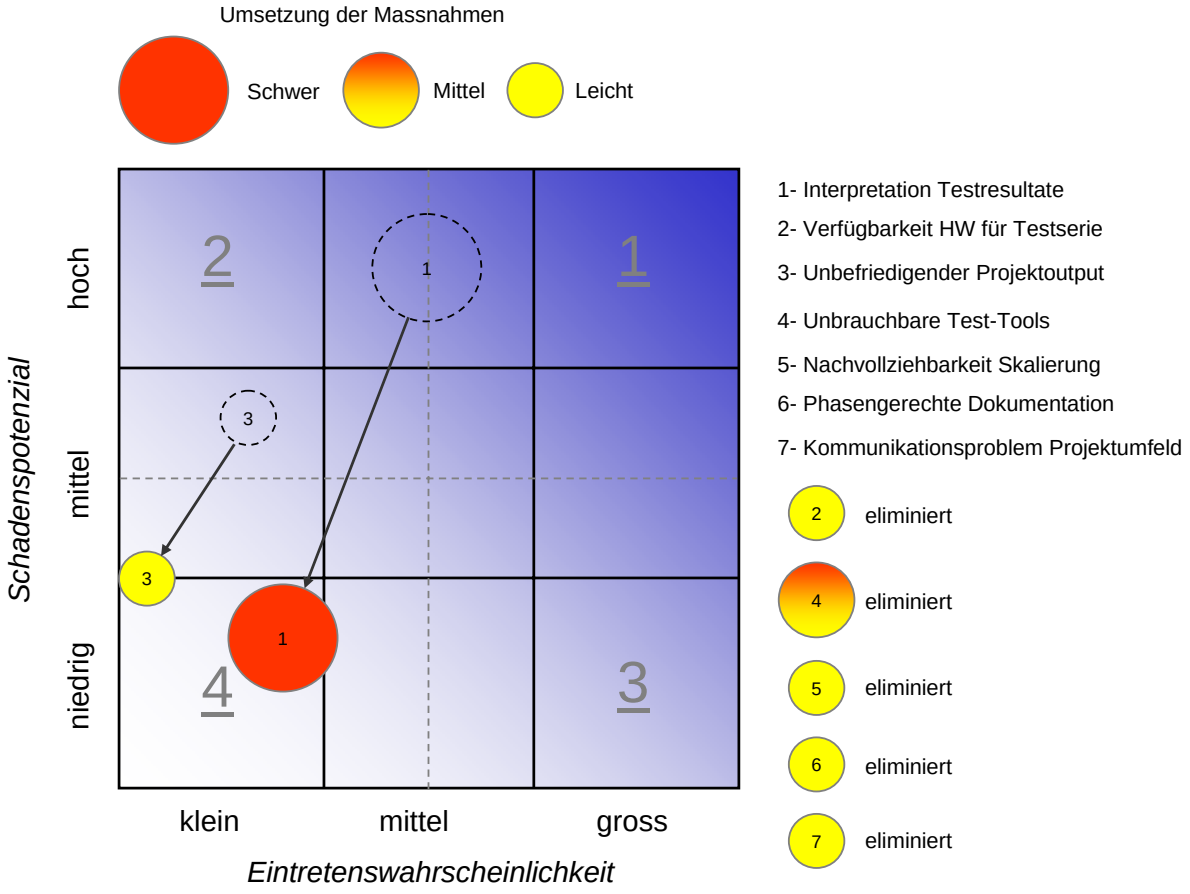

Abbildung 2 Risikobewertung

Tabelle 6 Risikoüberprüfung

#	Risiko	Status
1	**Interpretation Testresultate**	reduziert
	Mit zusätzlichen Analysetools wie bspw. CodeAnalyst von AMD konnten spezifische Zusammenhänge besser analysiert werden. Die Reproduzierbarkeit ist durch Testcases und –protokolle sichergestellt.	
2	**Verfügbarkeit HW für Testserie**	eliminiert
	Die Firma Pilatus Ltd. in Stans stellte für die Dauer der DA ein SMP-System zu Verfügung mit dem die ganze Testreihe durchgeführt werden konnte.	
3	**Unbefriedigender Projektoutput**	reduziert
	In der ersten Phase des Projekts wurde ein Zielkatalog erstellt und vom Dozent/Experten freigegeben. Mit der Basisanalyse wird das Thema Skalierung einer Applikation in mehreren Bereichen umfassend beleuchtet.	

4 **Unbrauchbare Test-Tools** eliminiert

Alle definierten Performance Indikatoren der Testcases konnten über die Konsolen-Ausgabe der Mandelbrot-Anwendung oder mit 2 frei zugänglichen Analysetools abgedeckt werden.

5 **Nachvollziehbarkeit Skalierung** eliminiert

Die Mandelbrot-Anwendung ermöglicht die visuelle Darstellung der Skalierung indem der Bildaufbau durch Worker-Threads sichtbar gemacht wird. Weiter zeigt die grafische Darstellung der Messresultate charakteristische Merkmale.

6 **Phasengerechte Dokumentation** eliminiert

Der Dokumentationsaufwand wurde in jeder Phase des Projektes eingeplant. Die laufende Aktualisierung/Erweiterung der Dokumente und vereinbarte Zwischenresultate führten zu einer umfassenden und qualitativ hochstehenden Projektdokumentation.

7 **Kommunikationsproblem Projektumfeld** eliminiert

Die Kommunikation mit Dozent, Experten und Auftraggeber stellte kein Problem dar. Mit Statusberichten, Projektreviews und Protokollen konnte der Informationsfluss sichergestellt werden.

6. Zielerreichung

Der im Software Project Management Plan SPMP definierte Zielkatalog soll in Bezug auf die Zielerreichung überprüft werden. Für Technische- und Methodische Zielsetzungen soll kritisch hinterfragt werden, ob die vorab definierten Erwartungen erfüllt werden konnten.

6.1.1. Technische Zielsetzungen

Tabelle 7 Technische Zielsetzungen

#	Zielsetzung	Kategorie	Status
{Z1}	Relevante Begriffe der Skalierung erläutern.	Muss	100
{Z2}	Relevante Grundlagen der Skalierung erläutern (bspw. Prozesse, Threads).	Muss	100
{Z3}	Bedingungen zur parallelen Abarbeitung einer Aufgabe aufzeigen.	Muss	80
{Z4}	Einsatzgebiete paralleler Anwendungen aufzeigen (bspw. Client, Server).	Muss	100
{Z5}	Software-Skalierung auf Java-Ebene (Basics, VM Klassen).	Muss	100
{Z6}	Software-Skalierung auf C++ Ebene (Basics, Libraries).	Kann	60
{Z7}	Skalierungstechniken auf Windows XP Ebene aufzeigen.	Muss	100
{Z8}	Skalierungstechniken auf Unix Ebene aufzeigen.	Kann	20
{Z9}	Skalierbare Prozessorarchitekturen aufzeigen (SMP).	Muss	100
{Z10}	Vergleich verschiedener Software-Plattformen.	Muss	100
{Z11}	Vergleich verschiedener Hardware-Plattformen.	Kann	100
{Z12}	Definition einer repräsentativen Testplattform (Hardware, OS, VM).	Muss	100
{Z13}	Testkonzept für verschiedene Designkonzepte definieren.	Muss	100
{Z14}	Suche und Evaluation von Test-Tools für skalierbare Anwendungen (Freeware).	Muss	100
{Z15}	Testklasse(n) zur Anwendung der verschiedenen Konzepte definieren.	Muss	100
{Z16}	Analyse der Konzepte anhand der Testklasse(n) auf Java-Basis (Implementierung).	Muss	100
{Z17}	Analyse der Konzepte anhand der Testklasse(n) auf C++-Basis (Implementierung).	Kann	0
{Z18}	Direkter Vergleich mehrerer Konzepte auf einer Plattform durchführen.	Muss	100
{Z19}	Direkter Vergleich der Plattformen mit einem ausgewählten Konzept durchführen.	Kann	100
{Z20}	Formulierung von Implementierungsempfehlungen (inkl. Code-Beispiele).	Muss	100

Hinweis zu {Z3}

Grundlagen werden vorausgesetzt, konkrete Umsetzung von Parallelisierung wurde anhand verschiedener Konzepte/Technologien erklärt (OpenMP, JOMP, TBB, Java Threads, POSIX Threads)

Hinweis zu {Z6}

Innerhalb der Basisanalyse wurden spezifische Themen auch in C/C++ beleuchtet. So beinhaltet diese Dokument Erklärungen zur Win32 API (Prozesse, Threads, Affinität), POSIX-Threads, OpenMP und TBB.

Hinweis zu {Z8}

Einige der vorgestellten Technologien lassen sich auch unter UNIX anwenden

Hinweis zu {Z17}

Es wurden keine Implementierungen und Tests im Bereich C/C++ vorgenommen. Konzepte wurden sprachübergreifend beschrieben und können so einfach auf andere Sprachen übertragen werden. Punktuell wurden Hinweise platziert wo eine vertiefte Analyse und Umsetzung in einer anderen Sprache sinnvoll wären.

6.1.2. Methodische Zielsetzungen

Tabelle 8 Methodische Zielsetzungen

#	Zielsetzung	Kategorie	Status
{Z21}	Inhalt der Teilprojektphasen durch Freigaben abgrenzen (Reviews, Meilensteine).	Muss	100
{Z22}	Projektdokumentation gemäss IEEE.	Muss	100
{Z23}	Projektplanung erstellen und permanente Fortschrittskontrolle.	Muss	100
{Z24}	Projektplanung durch Meilensteine strukturieren.	Muss	100
{Z25}	Implementierte Konzepte auf einer repräsentativen Plattform testen	Muss	100
{Z26}	Projektfortschritt mit Statusberichten dokumentieren.	Muss	100
{Z27}	Projektrisiken analysieren und dokumentieren.	Muss	100

7. Fazit und Erfahrungen

Thema

Ohne uns mit dem Thema der „Java Thread Skalierung“ im Voraus befasst zu haben, haben wir diese Themenstellung als mögliche Arbeit für unsere Diplomarbeit eingegeben. Die Internetrecherchen in der Basisanalyse haben uns schnell vor Augen geführt, wie aktuell diese Themenstellung im Kontext der steigenden Anzahl an Multi-Core und Multi-CPU Systemen ist.

Wir haben uns in der Startphase des Projektes schnell auf die vertikale Skalierung festgelegt und entschieden, dass Einflussfaktoren der Skalierung in jedem Layer eines Computersystems zu identifizieren und analysieren sind. Die Problemstellung der Abgrenzung drängte sich hierbei schnell auf, weil jeder Layer für sich ein komplexes Teilsystem darstellt in dem beliebig tief „gebohrt“ werden kann. Performante Java Programmierung oder das Scheduling des Betriebssystems sind Problemstellungen, die ausreichen würden um eine Diplomarbeit zu füllen.

Neben der Aufgabenstellung der HTA, wurden mit Projektzielsetzungen im SPMP die Themenschwerpunkte klar festgelegt. Diese „Guideline“ stellte im Voraus sicher, dass die Skalierung einer Applikation in der geforderten Breite beleuchtet wird und der Detaillierungsgrad im jeweiligen Layer überschaubar blieb.

Planung/Umsetzung

Die detaillierte Planung der Diplomarbeit in der Startphase des Projektes gab dieser Aufgabenstellung die themenspezifische und zeitliche Struktur für die Erarbeitung der geforderten Resultate. Durch die frühe Auseinandersetzung mit Projektphasen und deren Teilresultaten konnte eine realistische Abschätzung getroffen werden, was im Rahmen dieser Arbeit erreicht werden kann und was nicht. Das Ergebnis dieser Überlegungen ist in die oben beschriebenen Projektzielsetzungen eingeflossen.

Der geplante Arbeitsaufwand betrug für beide Dipolmanden in Summe 420 Std. Mit total 441 Std. haben wir das Ziel, ein brauchbares Resultat zu liefern und dabei die Abweichung so gering wie möglich zu halten, erreicht.

Team

Wird rückblickend das Team betrachtet, kann festgehalten werden, dass keine nennenswerten Probleme aufgetaucht sind, welche die Teamleistung geschwächt haben. Die Zusammenarbeit war über den ganzen Projektverlauf konstruktiv, zielorientiert und ermöglichte so eine effektive Verteilung der Arbeiten und ein effizientes Fortschreiten im Projekt. Die optimale Zusammenarbeit in der Jahresarbeit konnte in dieser Diplomarbeit wiederholt werden. Die Stärken unseres Teams lagen in den Erfahrungen im Management solcher Projekte, der (gemeinsamen) Arbeitstechnik, dem Problemlösungsprozess und dem Ehrgeiz im vorgegebenen Rahmen die gesteckten Zielsetzungen zu erfüllen.

Resultat/Dokumentation

Bis auf wenige Ausnahmen sind die Technologischen- und Methodischen-Zielsetzungen des Zielkataloges im SPMP zu 100% erreicht. Implementierungen von Konzepten in C++ (KANN-Ziel) und Skalierungstechniken auf Unix Ebene (KANN-Ziel) sind beispielsweise Themen die nicht oder nur in begrenztem Rahmen bearbeitet wurden. Der Grund führ diese fehlende Realisierung lag darin, dass fehlende Vorkenntnisse zu einem unverhältnismässigen Aufwand geführt hätten und darum gezielt vermieden wurden.

Im Zusammenhang mit der Festlegung der Schwerpunkte der Aufgabenstellung (Sicht Projektteam) im SPMP ergibt sich ebenfalls ein erfreuliches Bild. Die Arbeitsaufwände für Basisanalyse, Evaluation oder Umsetzung/Implementierung, etc. lagen im Rahmen der Planwerte. Für die Dokumentation dieser Arbeit wurde aber deutlich mehr Zeit investiert als vorgesehen. Dies führte einerseits zu einer umfassenden Gesamtdokumentation mit diversen Teildokumenten und andererseits zum Überschreiten des Stunden-Budgets für diese Arbeit. Wir haben als Team einen grösseren Nutzen darin gesehen, eine durchgängige, aussagekräftige Dokumentation dieser Arbeit zu liefern.

8. Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbstständig und unter ausschliesslicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Horw, 24. November 2006

Marcel Aregger

Rainer Meier

Journey through Mandelbrot