APL-Journal

A Programming Language

2/2011

Frank Schmidt-Kelletat
Erzeugen von Druckoutputs

Ralf Herminghaus
APL-Benchmark: Wie schnell ist APL wirklich?

Wolfgang Hengstler
Exploring the Origins of IBM DB2 Analytics Accelerator

Peter Thaben
Warum eigentlich Linux?

Im Blickpunkt

Jürgen Hanneder
Der TeXnische Fortschritt und seine Tücken

APL-Germany e.V.
Jahrgang 30 Nr. 2 2011
ISSN - 1438-4531
RHOMBOS-VERLAG

APL-Journal

30. Jg. 2011, ISSN 1438-4531

Herausgeber: Dr. Reiner Nussbaum, APL-Germany e.V., Mannheim, http://www.apl-germany.de

Redaktion: Dipl.-Volksw. Martin Barghoorn (verantw.), TU Berlin, Franklinstr. 28, D-10587 Berlin, Tel. (030) 314 24392, Fax (030) 314 25901

Verlag: RHOMBOS-VERLAG, Berlin, Kurfürstenstr. 17, D-10785 Berlin, Tel. (030) 261 9461, Fax (030) 261 6300, eMail: verlag@rhombos.de, Internet: www.rhombos.de

Erscheinungsweise: halbjährlich

Erscheinungsort: Berlin

Satz: Rhombos-Verlag

Druck: dbusiness.de GmbH, Berlin

Copyright: APL Germany e.V. (für alle Beiträge, die als Erstveröffentlichung erscheinen)

Fotonachweis Titelseite: Martin Barghoorn

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutzgesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen. Eine Haftung für die Richtigkeit der veröffentlichten Informationen kann trotz sorgfältiger Prüfung von Herausgeber und Verlag nicht übernommen werden. Mit Namen gekennzeichnete Artikel geben nicht unbedingt die Meinung des Herausgebers oder der Redaktion wieder. Für unverlangte Einsendungen wird keine Haftung übernommen. Nachdruck ist nur mit Zustimmung des Herausgebers sowie mit Quellenangabe und Einsendung eines Beleges gestattet. Überarbeitungen eingesandter Manuskripte liegen im Ermessen der Redaktion.

Liebe APL-Freunde,

es ist mir eine Freude, Ihnen das vorliegende APL-Journal zukommen zu lassen.

Im Jahr 2011 hatten wir zwei Tagungen: Die Frühjahrstagung fand in Köln bei der Deutschen Krankenversicherung (DKV) statt und die Herbsttagung bei der Allianz Deutschland AG in Stuttgart. Den Gastgebern der Veranstaltungen sei an dieser Stelle noch einmal herzlich gedankt.

Viel Vergnügen bei dem Lesen dieser APL-Ausgabe wünscht Ihnen

Ihr Reiner Nussbaum

P.S.: Im Frühjahr 2013 finden bei APL-Germany e.V. Vorstandswahlen statt. Insbesondere für die Positionen Kassenwart und Schriftführer streben die derzeitigen Vorstandsmitglieder eine personelle Neubesetzung an. Interessierte Mitglieder sind aufgerufen, sich für diese Positionen zu engagieren.

INHALT

Frank Schmidt-Kelletat

Erzeugen von Druckoutputs

Damit Anwender von IBM-APL2 basierten Applikationen attraktive Druckergebnisse erzielen können, sollte ein Mindestmaß an Gestaltungsmitteln wie zum Beispiel die Nutzung von Farben oder unterschiedlicher Zeichenformatierungen Standard sein. Das technische Fundament für wahlfreie Gestaltungen stellt die externe Funktion COM dar, die den Zugriff auf die Microsoft Produkte und damit auf ihre bekannten Formatierungsmöglichkeiten erlaubt.

Ralf Herminghaus

APL-Benchmark: Wie schnell ist APL wirklich?

Zwar scheiterte der Versuch, die Geschwindigkeiten unterschiedlicher APL-Interpreter miteinander zu vergleichen und eventuell sogar den schnellsten und besten der auf dem Markt angebotenen Interpreter herauszufinden. Die Beschäftigung mit Benchmarks im Allgemeinen und den Einzelergebnissen der hier vorgelegten Untersuchung im Besonderen fördert trotzdem einige interessante und teilweise überraschende Erkenntnisse zutage.

Wolfgang Hengstler

Exploring the Origins of IBM DB2 Analytics Accelerator

Given the phenomenal response to the IBM DB2 Analytics Accelerator, the IBM Data Management Magazine is taking an in-depth look into the product and its origins.

Peter Thaben

Warum eigentlich Linux?

Die ganze Welt ist vom Betriebssystem Windows besetzt … Die ganze Welt? Nein! Eine kleine unbeugsame Gemeinde von Mainstreamverweigerern frönt einer exotischen Leidenschaft: Sie benutzt Linux und - man mag es kaum laut sagen - sie ist glücklich damit und vermisst nichts.

Jürgen Hanneder

Der TEXnische Fortschritt und seine Tücken

Am Beispiel der Indischen Sprachen berichtet ein TEX-Anwender der ersten Stunde über die beachtlichen Fortschritte, die dieses Textsatzsystem mit eingebauter Makrosprache gemacht hat, und über die Tücken und Unzulänglichkeiten, die dennoch in der Praxis auftauchen.

Erzeugen von Druckoutputs 3

APL-Benchmark: Wie schnell ist APL wirklich? 11

Exploring the Origins of IBM DB2 Analytics Accelerator 25

Warum eigentlich Linux? 31

Der TEXnische Fortschritt und seine Tücken 40

Nachrichten 46

Impressum 48

Bildnachweis:
Martin Barghoorn (Umschlagseite 1)

Frank Schmidt-Kelletat

Erzeugen von Druckoutputs

Möchte man heutzutage den Anwendern von IBM-APL2 basierten Applikationen attraktive Druckergebnisse anbieten wollen, dann würde es nicht ausreichen, lediglich reine ASCII-Zeichen auf das Papier zu bringen, sondern ein Mindestmaß an Gestaltungsmitteln wie zum Beispiel die Nutzung von Farben oder unterschiedlicher Zeichenformatierungen sollte Standard sein. Das technische Fundament für wahlfreie Gestaltungen stellt die externe Funktion COM dar, die den Zugriff auf die Microsoft Produkte und damit auf ihre bekannten Formatierungsmöglichkeiten erlaubt.

Der Startschuss bei der Programmierung eines optisch attraktiven Druckbildes fällt mit der Entscheidung des Entwicklers darüber, in welcher Applikation (Anmerk.: die nachfolgende Ausführungen konzentrieren sich dabei auf Excel) der Druck ausgeführt und damit sinnigerweise auch die Gestaltung vorgenommen werden soll. Daran schließt sich die Frage an, wie eine technische Lösung aussehen kann. Hierbei stellt die Nutzung von Visual Basic-Anweisungen den einfachsten Weg dar, weil in den Office-Produkten ein Mechanismus integriert ist, der es erlaubt, Code aufzuzeichnen der in APL weiterverwendet werden kann.

Vorbereitung

Die externe Funktion COM assoziiert man durch folgende Befehlseingabe:

```
      3 11 DNA 'COM'
```

Mit Hilfe des integrierten Dokumentes User's Guide ist bekannt, dass die Applikation Excel durch folgende Synatx gestartet wird:

```
H¯EXCEL←COM 'CREATE' 'EXCEL.APPLICATION'
```

Da Excel standardmäßig unsichtbar und ohne Workbook eröffnet wird, schließen sich noch weitere zwei Befehle an:

```
COM ⊤PROPERTY H¯EXCEL ⊤VISIBLE   1
H¯BOOK←COM ⊤METHOD  H¯EXCEL ⊤Workbooks.Add
```

Die Vorbereitungen sind abgeschlossen, wenn die Beispieldaten MAT durch folgende Anweisung an Excel übergeben worden sind:

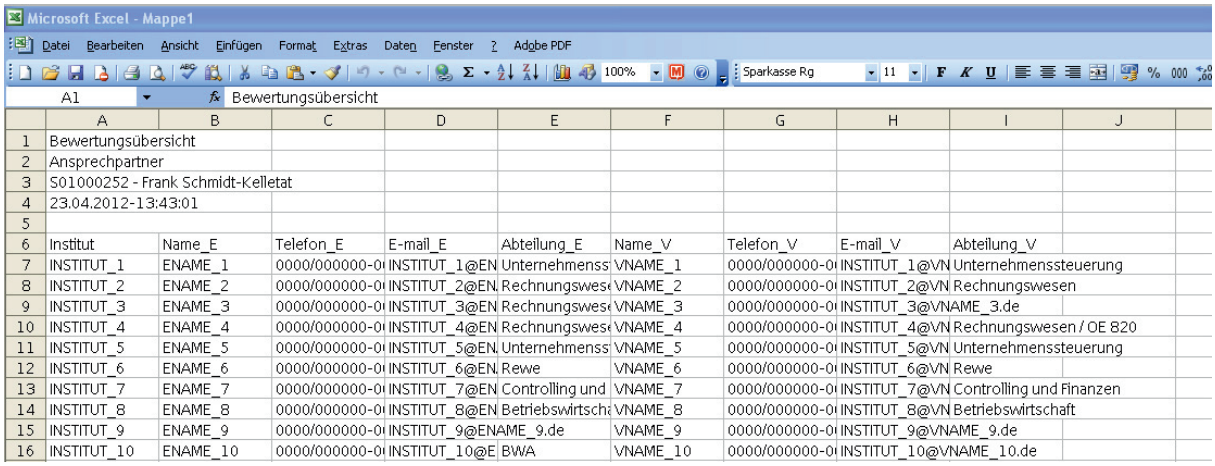
```
COM 'PROPERTY' H¯EXCEL 'CELLS().RES IZE().
VALUE'(1 1)(⍴MAT)(MAT)
```

Die Darstellung in Excel bietet nun das nachfolgende Bild (siehe Abbildung 1).

Einsatz von Visual Basic

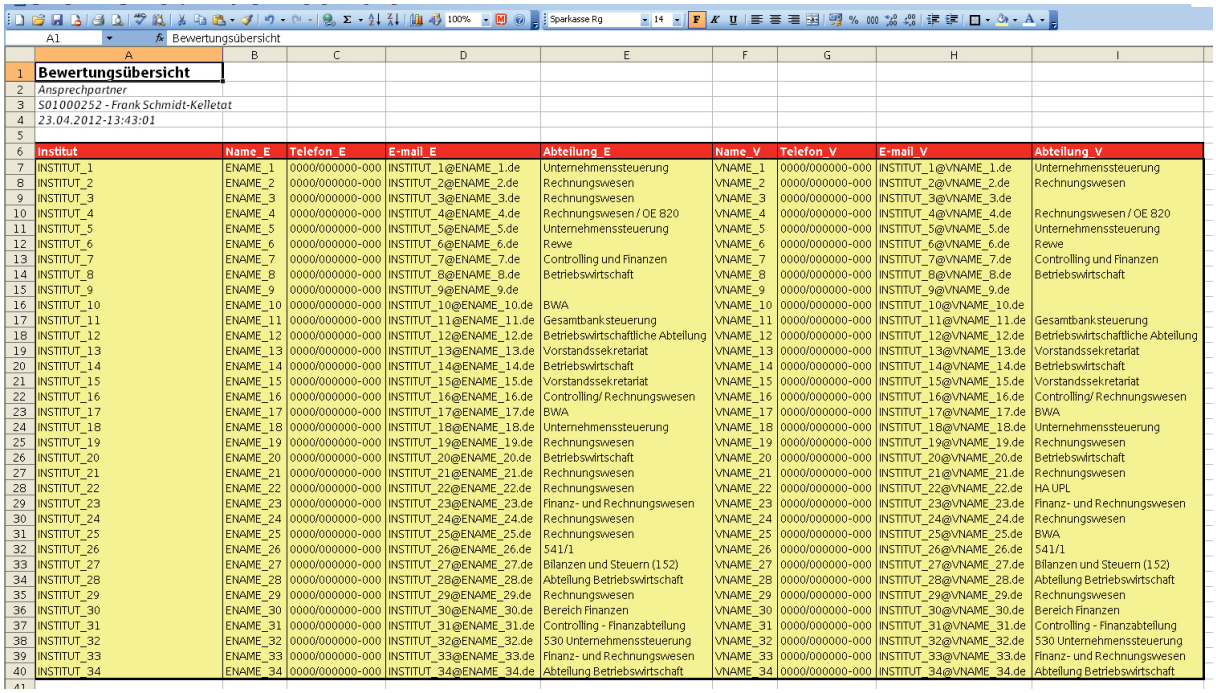
Nun ist der Gestaltungsfreiheit keine Grenze gesetzt. Um die nun folgenden verschiedenen Einstellungen dauerhaft nutzen zu können, sollen sie protokolliert werden – dazu wird der Makro-Rekorder gestartet und die Formatierung kann beginnen:

Zunächst einmal sollen alle relevanten Spalten auf eine optimale Spaltenbreite eingestellt werden. Daran anschließend wird die Zelle A1 fett und in der Schriftgröße 14, die Zellen A2 bis A4 kursiv formatiert. Bei der spaltenorientierten Anordnung der relevanten Informationen eignet sich zur optischen Untermalung die Verwendung eines Rahmens. Man markiert das Gebilde und wählt für die äußeren Grenzen fette und für die inneren vertikalen Rahmen dünne Linien. Die Kopfzeilen der Spalten kann man ebenfalls mit einem mittleren Rahmentypen versehen, den Hintergrund der Kopfzeilen beispielsweise rot färben und die Schriftfarbe auf weiß wechseln. Der Body der Tabelle wird in diesem Beispiel hellgelb gefärbt – danach soll die Formatierung abgeschlossen sein und die Makroaufzeichnung beendet werden. Die Darstellung kann Abbildung 2 entnommen werden.



	A	B	C	D	E	F	G	H	I	J
1	Bewertungsübersicht									
2	Ansprechpartner									
3	S01000252 - Frank Schmidt-Kelletat									
4	23.04.2012-13:43:01									
5										
6	Institut	Name_E	Telefon_E	E-mail_E	Abteilung_E	Name_V	Telefon_V	E-mail_V	Abteilung_V	
7	INSTITUT_1	ENAME_1	0000/000000-0	INSTITUT_1@EN	Unternehmenss	VNAME_1	0000/000000-0	INSTITUT_1@VN	Unternehmenssteuerung	
8	INSTITUT_2	ENAME_2	0000/000000-0	INSTITUT_2@EN	Rechnungswes	VNAME_2	0000/000000-0	INSTITUT_2@VN	Rechnungswesen	
9	INSTITUT_3	ENAME_3	0000/000000-0	INSTITUT_3@EN	Rechnungswes	VNAME_3	0000/000000-0	INSTITUT_3@VNAME_3.de		
10	INSTITUT_4	ENAME_4	0000/000000-0	INSTITUT_4@EN	Rechnungswes	VNAME_4	0000/000000-0	INSTITUT_4@VN	Rechnungswesen / OE 820	
11	INSTITUT_5	ENAME_5	0000/000000-0	INSTITUT_5@EN	Unternehmenss	VNAME_5	0000/000000-0	INSTITUT_5@VN	Unternehmenssteuerung	
12	INSTITUT_6	ENAME_6	0000/000000-0	INSTITUT_6@EN	Rewe	VNAME_6	0000/000000-0	INSTITUT_6@VN	Rewe	
13	INSTITUT_7	ENAME_7	0000/000000-0	INSTITUT_7@EN	Controlling und	VNAME_7	0000/000000-0	INSTITUT_7@VN	Controlling und Finanzen	
14	INSTITUT_8	ENAME_8	0000/000000-0	INSTITUT_8@EN	Betriebswirtsch	VNAME_8	0000/000000-0	INSTITUT_8@VN	Betriebswirtschaft	
15	INSTITUT_9	ENAME_9	0000/000000-0	INSTITUT_9@ENAME_9.de		VNAME_9	0000/000000-0	INSTITUT_9@VNAME_9.de		
16	INSTITUT_10	ENAME_10	0000/000000-0	INSTITUT_10@E	BWA	VNAME_10	0000/000000-0	INSTITUT_10@VNAME_10.de		

Abb. 1: Excel-Darstellung vor Formatierung



	A	B	C	D	E	F	G	H	I
1	**Bewertungsübersicht**								
2	*Ansprechpartner*								
3	*S01000252 - Frank Schmidt-Kelletat*								
4	*23.04.2012-13:43:01*								
5									
6	**Institut**	**Name_E**	**Telefon_E**	**E-mail_E**	**Abteilung_E**	**Name_V**	**Telefon_V**	**E-mail_V**	**Abteilung_V**
7	INSTITUT_1	ENAME_1	0000/000000-000	INSTITUT_1@ENAME_1.de	Unternehmenssteuerung	VNAME_1	0000/000000-000	INSTITUT_1@VNAME_1.de	Unternehmenssteuerung
8	INSTITUT_2	ENAME_2	0000/000000-000	INSTITUT_2@ENAME_2.de	Rechnungswesen	VNAME_2	0000/000000-000	INSTITUT_2@VNAME_2.de	Rechnungswesen
9	INSTITUT_3	ENAME_3	0000/000000-000	INSTITUT_3@ENAME_3.de	Rechnungswesen	VNAME_3	0000/000000-000	INSTITUT_3@VNAME_3.de	
10	INSTITUT_4	ENAME_4	0000/000000-000	INSTITUT_4@ENAME_4.de	Rechnungswesen / OE 820	VNAME_4	0000/000000-000	INSTITUT_4@VNAME_4.de	Rechnungswesen / OE 820
11	INSTITUT_5	ENAME_5	0000/000000-000	INSTITUT_5@ENAME_5.de	Unternehmenssteuerung	VNAME_5	0000/000000-000	INSTITUT_5@VNAME_5.de	Unternehmenssteuerung
12	INSTITUT_6	ENAME_6	0000/000000-000	INSTITUT_6@ENAME_6.de	Rewe	VNAME_6	0000/000000-000	INSTITUT_6@VNAME_6.de	Rewe
13	INSTITUT_7	ENAME_7	0000/000000-000	INSTITUT_7@ENAME_7.de	Controlling und Finanzen	VNAME_7	0000/000000-000	INSTITUT_7@VNAME_7.de	Controlling und Finanzen
14	INSTITUT_8	ENAME_8	0000/000000-000	INSTITUT_8@ENAME_8.de	Betriebswirtschaft	VNAME_8	0000/000000-000	INSTITUT_8@VNAME_8.de	Betriebswirtschaft
15	INSTITUT_9	ENAME_9	0000/000000-000	INSTITUT_9@ENAME_9.de		VNAME_9	0000/000000-000	INSTITUT_9@VNAME_9.de	
16	INSTITUT_10	ENAME_10	0000/000000-000	INSTITUT_10@ENAME_10.de	BWA	VNAME_10	0000/000000-000	INSTITUT_10@VNAME_10.de	
17	INSTITUT_11	ENAME_11	0000/000000-000	INSTITUT_11@ENAME_11.de	Gesamtbanksteuerung	VNAME_11	0000/000000-000	INSTITUT_11@VNAME_11.de	Gesamtbanksteuerung
18	INSTITUT_12	ENAME_12	0000/000000-000	INSTITUT_12@ENAME_12.de	Betriebswirtschaftliche Abteilung	VNAME_12	0000/000000-000	INSTITUT_12@VNAME_12.de	Betriebswirtschaftliche Abteilung
19	INSTITUT_13	ENAME_13	0000/000000-000	INSTITUT_13@ENAME_13.de	Vorstandssekretariat	VNAME_13	0000/000000-000	INSTITUT_13@VNAME_13.de	Vorstandssekretariat
20	INSTITUT_14	ENAME_14	0000/000000-000	INSTITUT_14@ENAME_14.de	Betriebswirtschaft	VNAME_14	0000/000000-000	INSTITUT_14@VNAME_14.de	Betriebswirtschaft
21	INSTITUT_15	ENAME_15	0000/000000-000	INSTITUT_15@ENAME_15.de	Vorstandssekretariat	VNAME_15	0000/000000-000	INSTITUT_15@VNAME_15.de	Vorstandssekretariat
22	INSTITUT_16	ENAME_16	0000/000000-000	INSTITUT_16@ENAME_16.de	Controlling/ Rechnungswesen	VNAME_16	0000/000000-000	INSTITUT_16@VNAME_16.de	Controlling/ Rechnungswesen
23	INSTITUT_17	ENAME_17	0000/000000-000	INSTITUT_17@ENAME_17.de	BWA	VNAME_17	0000/000000-000	INSTITUT_17@VNAME_17.de	BWA
24	INSTITUT_18	ENAME_18	0000/000000-000	INSTITUT_18@ENAME_18.de	Unternehmenssteuerung	VNAME_18	0000/000000-000	INSTITUT_18@VNAME_18.de	Unternehmenssteuerung
25	INSTITUT_19	ENAME_19	0000/000000-000	INSTITUT_19@ENAME_19.de	Rechnungswesen	VNAME_19	0000/000000-000	INSTITUT_19@VNAME_19.de	Rechnungswesen
26	INSTITUT_20	ENAME_20	0000/000000-000	INSTITUT_20@ENAME_20.de	Betriebswirtschaft	VNAME_20	0000/000000-000	INSTITUT_20@VNAME_20.de	Betriebswirtschaft
27	INSTITUT_21	ENAME_21	0000/000000-000	INSTITUT_21@ENAME_21.de	Rechnungswesen	VNAME_21	0000/000000-000	INSTITUT_21@VNAME_21.de	Rechnungswesen
28	INSTITUT_22	ENAME_22	0000/000000-000	INSTITUT_22@ENAME_22.de	Rechnungswesen	VNAME_22	0000/000000-000	INSTITUT_22@VNAME_22.de	HA UPL
29	INSTITUT_23	ENAME_23	0000/000000-000	INSTITUT_23@ENAME_23.de	Finanz- und Rechnungswesen	VNAME_23	0000/000000-000	INSTITUT_23@VNAME_23.de	Finanz- und Rechnungswesen
30	INSTITUT_24	ENAME_24	0000/000000-000	INSTITUT_24@ENAME_24.de	Rechnungswesen	VNAME_24	0000/000000-000	INSTITUT_24@VNAME_24.de	Rechnungswesen
31	INSTITUT_25	ENAME_25	0000/000000-000	INSTITUT_25@ENAME_25.de	Rechnungswesen	VNAME_25	0000/000000-000	INSTITUT_25@VNAME_25.de	BWA
32	INSTITUT_26	ENAME_26	0000/000000-000	INSTITUT_26@ENAME_26.de	541/1	VNAME_26	0000/000000-000	INSTITUT_26@VNAME_26.de	541/1
33	INSTITUT_27	ENAME_27	0000/000000-000	INSTITUT_27@ENAME_27.de	Bilanzen und Steuern (152)	VNAME_27	0000/000000-000	INSTITUT_27@VNAME_27.de	Bilanzen und Steuern (152)
34	INSTITUT_28	ENAME_28	0000/000000-000	INSTITUT_28@ENAME_28.de	Abteilung Betriebswirtschaft	VNAME_28	0000/000000-000	INSTITUT_28@VNAME_28.de	Abteilung Betriebswirtschaft
35	INSTITUT_29	ENAME_29	0000/000000-000	INSTITUT_29@ENAME_29.de	Rechnungswesen	VNAME_29	0000/000000-000	INSTITUT_29@VNAME_29.de	Rechnungswesen
36	INSTITUT_30	ENAME_30	0000/000000-000	INSTITUT_30@ENAME_30.de	Bereich Finanzen	VNAME_30	0000/000000-000	INSTITUT_30@VNAME_30.de	Bereich Finanzen
37	INSTITUT_31	ENAME_31	0000/000000-000	INSTITUT_31@ENAME_31.de	Controlling - Finanzabteilung	VNAME_31	0000/000000-000	INSTITUT_31@VNAME_31.de	Controlling - Finanzabteilung
38	INSTITUT_32	ENAME_32	0000/000000-000	INSTITUT_32@ENAME_32.de	530 Unternehmenssteuerung	VNAME_32	0000/000000-000	INSTITUT_32@VNAME_32.de	530 Unternehmenssteuerung
39	INSTITUT_33	ENAME_33	0000/000000-000	INSTITUT_33@ENAME_33.de	Finanz- und Rechnungswesen	VNAME_33	0000/000000-000	INSTITUT_33@VNAME_33.de	Finanz- und Rechnungswesen
40	INSTITUT_34	ENAME_34	0000/000000-000	INSTITUT_34@ENAME_34.de	Abteilung Betriebswirtschaft	VNAME_34	0000/000000-000	INSTITUT_34@VNAME_34.de	Abteilung Betriebswirtschaft

Abb. 2: Excel-Darstellung nach Formatierung

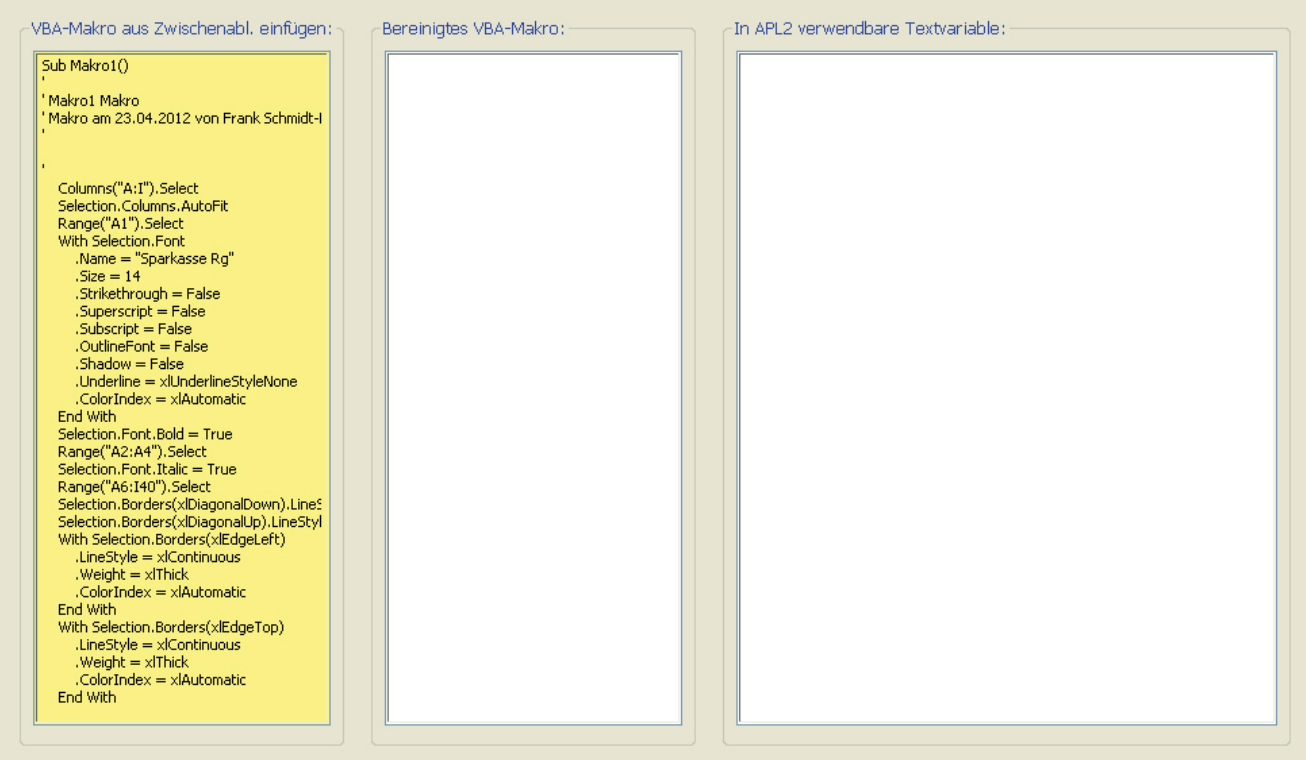


Abb. 3: Werkzeug zur syntaktischen Aufbereitung

Das entstandene Makro lässt sich über den Visual Basic-Editor einsehen. Für die weitere Verarbeitung markiert und kopiert man es in die Zwischenablage. Excel kann man nun schließen, da es an dieser Stelle nicht mehr für die Weiterentwicklung benötigt wird. Die findet nun im APL — konkret in einem eigens konstruierten Werkzeug — statt. Hier wird der VBA-Code eingefügt und automatisch für die Verwendung in APL syntaktisch aufbereitet (siehe Abbildung 3).

Beim Betrachten des VBA-Codes fällt zunächst auf, dass einige Zeilen Standardeinträge enthalten, die man für eine weitere Nutzung nicht benötigt. Dies betrifft alle Zeilen die auf „False“, „None“ oder „xlAutomatic“ enden. Alle Leer- und Kommentarzeilen sind ebenfalls vernachlässigbar. Deshalb werden im Beispiel zunächst diese Zeilen entfernt. Der Zwischenstand wird in der nachfolgenden Abbildung 4 dargestellt.

Dieser Zwischenstand birgt weiteres Optimierungspotential, denn wie man deutlich erkennen kann, hat man es mit immer wiederkehrenden WITH/END WITH Mustern zu tun. Diese Schreibweise findet in Visual Basic immer dann Berücksichtigung, wenn man für ein bestimmtes Objekt eine Reihe von Anweisungen ausführen möchte, ohne den Namen des Objekts immer wieder neu angeben zu müssen. Das bedeutet im Umkehrschluss, dass man auf die Struktur verzichten kann und damit jeweils zwei Zeilen einspart, wenn man bereit ist, den Namen des Objekts für jede zu vergebende Eigenschaft zu wiederholen. Da wir es hier mit einem Automatismus zu tun haben, lässt sich der Verzicht auf die WITH/END WITH-Struktur leicht

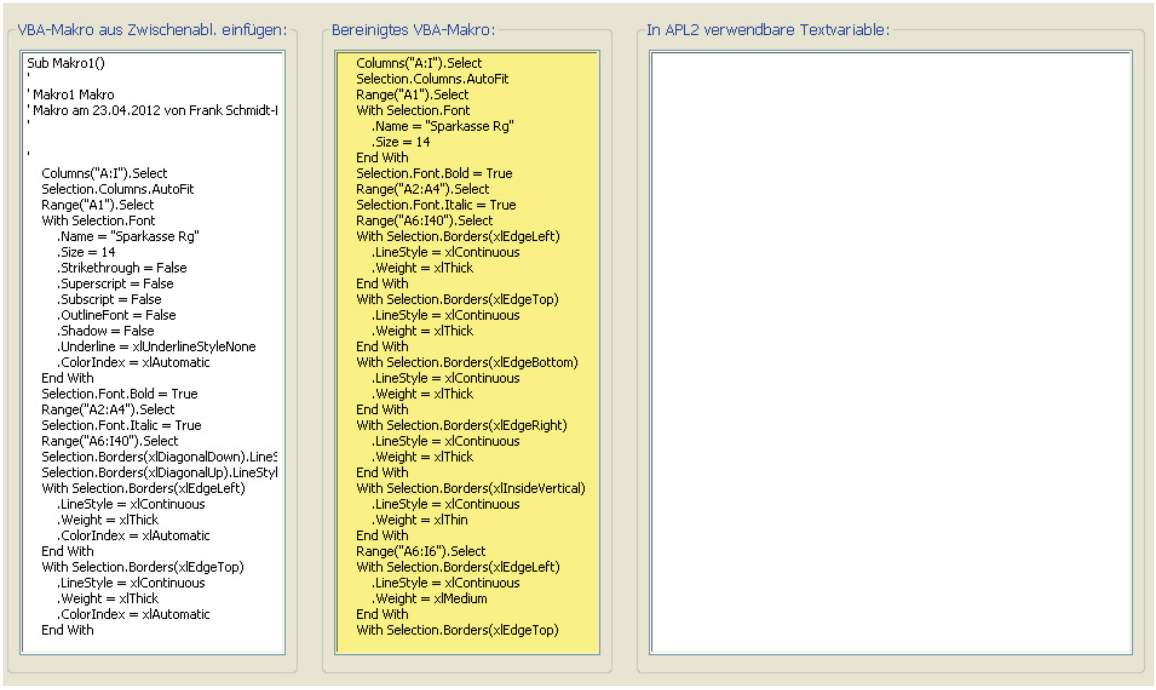


Abb. 4: Entfernung überflüssiger Zeilen

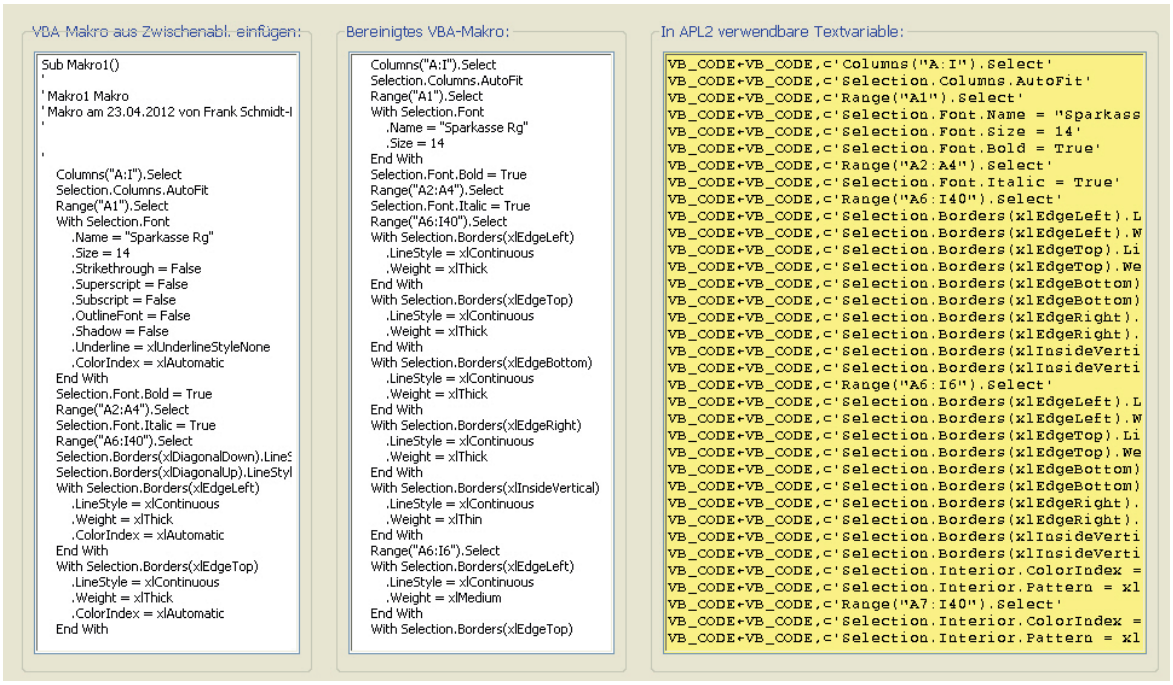


Abb. 5: Letzte Bereinigung und Zielformat

realisieren. Neben dieser letzten Bereinigung soll nun aber auch das Zielformat berücksichtigt werden, in dem die Befehle zur Formatierung in ein APL-Programm Eingang finden (siehe Abbildung 5).

Abbildung 5 demonstriert den Aufbau einer Variable VB_CODE. Jede verbliebene VBA-Anweisung wird der Variable als eingepackter String übergeben, wodurch man einen Vektor aus Textvektoren erhält. Diese

```
[13]  →0×⍴↑(RC H¯EXCEL H¯BOOK)←1(3⊃ΔL)0 EXCEL_INI ''
[14]
[15]  →L_E×⍴1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'CELLS().RESIZE().VALUE'(1 1)(⍴ΔR)(ΔR)
[16]  →L_E×⍴↑(RC MSG H¯VBC)←↑¨1 COM 'PROPERTY' H¯BOOK 'VBProject.VBComponents'
[17]  →L_E×⍴↑(RC MSG H¯COMP)←↑¨1 COM 'METHOD' H¯VBC 'ADD' 1
[18]  →L_E×⍴↑(RC MSG H¯CMOD)←↑¨1 COM 'PROPERTY' H¯COMP 'CodeModule'
[19]  →L_E×⍴1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'Worksheets().Activate' 1
[20]
[21]  VB_CODE←⊂'Sub Formatierung'
[22]  VB_CODE←VB_CODE,⊂'Application.ScreenUpdating = false'
[23]
[24] ⍝ Reservierter Raum für VBA-Makro
[25]
[26]  VB_CODE←VB_CODE,⊂'Range("A1").Select'
[27]  VB_CODE←VB_CODE,⊂'Application.ScreenUpdating = true'
[28]  VB_CODE←VB_CODE,⊂'End Sub'
[29]  →L_E×⍴↑↑1 COM 'METHOD' H¯CMOD 'InsertLines' 2(⍪VB_CODE,¨⊂⎕TC)
[30]  0 0⍴1 COM 'METHOD' H¯EXCEL 'RUN' 'Formatierung'
[31]
[32]  0 0⍴1 COM 'PROPERTY' H¯EXCEL 'ActiveSheet.PrintOut' 1
[33]
[34] L_E:
[35]  →L_REL×⍴3⊃ΔL
[36]  0 0⍴1 COM 'METHOD' H¯BOOK 'Close'
[37]  0 0⍴1 COM 'METHOD' H¯EXCEL 'Quit'
[38]
[39] L_REL:
[40]  COM_RELEASE 'H¯BOOK' 'H¯EXCEL' 'RC' 'H¯VBC' 'H¯COMP' 'H¯CMOD'
```

Abb. 6: Zielfunktion noch ohne Zwischenablageinhalt

Darstellung besteht aber bis hierhin nur auf dem Papier bzw. in einem MLE-Control. Um die Beschreibung zum Aufbau einer Variablen VB_CODE verwenden zu können, werden alle Zeilen markiert, in die Zwischenablage kopiert und in die Zielfunktion eingefügt (siehe Abbildung 6).

Die Rohform der Zielfunktion gibt die relevanten Anweisungen zur Datenübergabe und Formatierung wieder. Zeile [13] öffnet Excel sichtbar und mit Workbook, Zeile [14] übergibt die Daten. Um die oben beschriebene Technik zur Formatierung von Daten über ein VBA-Makro anwenden zu können, muss ein Modul in einem sogenannten VBA-Projekt der geöffneten Excel-Instanz anlegbar sein. Die Zeilen [16] bis [18] übernehmen diese Aufgabe. Zeile [19] stellt sicher, dass die erste Tabelle der eröffneten Arbeitsmappe aktiv ist. In Zeile [21] wird die schon erwähnte Variable VB_CODE generiert, in dem ein String zugewiesen wird, der dem Kopf des aufzubauenden Makros entspricht. Der in der Zeile [22] verkettete Ausdruck sorgt zur Laufzeit des Makros dafür, dass nicht jeder Formatierungsschritt am Bildschirm sichtbar wird, denn die Bildschirmaktualisierung soll erst zum Ende des Makrolaufs ausgeführt werden. Das lässt die Programmierung sehr viel ruhiger und damit professioneller wirken. Nun wird die in der Zwischenablage gespeicherte Formulierung der Textvariable VB_CODE an der kommentierten Stelle „Reservierter Raum für Makro“ eingefügt. Die Zeilen [26] bis [28] – Cursor auf Zelle A1 setzen, Bildschirm aktualisieren und Makrodefinition beenden - komplettieren den Aufbau der Variable VB_CODE. Danach ist die Funktion soweit ausgearbeitet, dass sie jederzeit das gewünschte Formatierungsergebnis im Excel produzieren kann. Die Zeile [29] sendet die Variable VB_CODE in der Struktur Carriage Return/Line Feed in das Excel-Modul, Zeile [30] erzwingt abschließend das Ausführen des Makros. Über eine

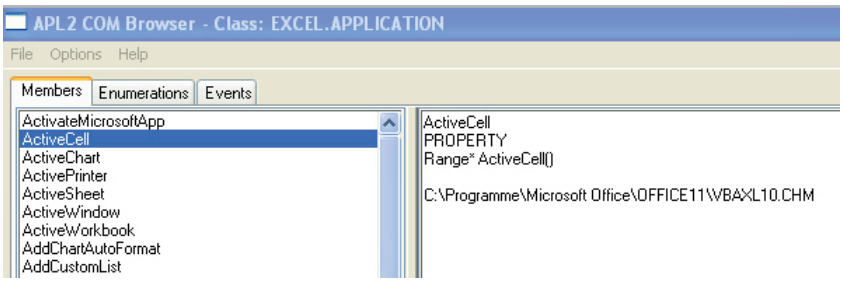


Abb. 7: Teilansicht von COMBROWSE

COM-Formulierung in Zeile [32] bewirkt APL, dass Excel das Worksheet ausdruckt. Zur Vollständigkeit sei noch erwähnt, dass in Abhängigkeit eines Argumentes in Zeile [35] gesteuert wird, ob Excel beendet wird. Zeile [40] gibt zum Abschluss der Funktion alle erzeugten Handle frei.

Direkte Excel-Kontrolle

Der kritische Leser wird bemerken, dass die Technik der Steuerung über ein Visual Basic—Makro nur dann realisierbar ist, wenn Excel im Bereich der Makrosicherheit den Zugriff auf Visual Basic Projekte erlaubt. Liegt diese Erlaubnis nicht vor, so muss die Steuerung aus dem APL heraus direkt erfolgen — die VBA-Syntax muss dementsprechend angepasst werden und es stellt sich die Frage, wie? Die Antwort lautet: alle Informationen über COM sammeln, die man bekommen kann! Ein ganz wichtiger Hinweisgeber verbirgt sich hinter der externen Funktion COMBROWSE, die man wie folgt einsetzen kann:

```
      3 11 DNA 'COMBROWSE'
1
      COMBROWSE 'EXCEL.APPLICATION'
```

Durch Verwenden der Funktion im Zusammenhang mit der Zielapplikation erhält man eine Registerblattdarstellung der Members, Enumerations und Events der zu steuernden Anwendung (Abbildung 7).

Über dieses Hilfsmittel kann man u.a. etwas über weitere Hilfedokumente erfahren, wie im abgebildeten Fall des Property ActiveCell im Pfad C:\Programme\Microsoft Office\Office11\ der lokalen Maschine — dort findet sich eine Hilfedatei mit der Bezeichnung VBAXL10.CHM. Der COMBROWSER enthält aber z. B. auch die Information darüber, wie folgende VBA-Anweisung umdefiniert wird:

```
VB←CODE←VB_CODE, ⊂'Selection.Borders
  (xlEdgeLeft).LineStyle = xlContinuous'
```

Die VBA-Formulierung sorgt dafür, dass „die linken Rahmen des markierten Bereiches mit einer durchgezogenen Linie“ formatiert werden. Die Elemente xlEdgeLeft und xlContinuous bezeichnet man als Constants. Sie leiten sich aus den Enumerations ab. Um diese Elemente für die direkte COM-Kontrolle aus APL nutzen zu können, benötigt man aber statt der Bezeichnungen der Constants deren Werte. Dafür muss man einen Blick in die Registerkarte Enumerations werfen und dort nach den Einträgen Borders und Linestyle suchen. Leider findet man beide gesuchten Begriffe in der Liste nur unter ähnlichen

Bezeichnungen: Borders entsprichet XlBordersIndex, Linestyle entspricht XlLineStyle. Wählt man XlBorders ergibt sich folgende Bildschirmansicht:

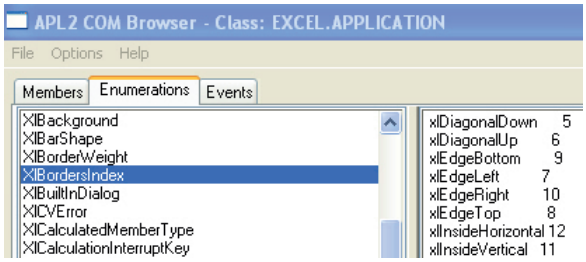


Abb. 8: Teilansicht der Enumerations in COMBROWSE

Hier findet man auf der rechten Seite die Constants-Bezeichnung xlEdgeLeft und ihren entsprechenden Wert 7. Für XlLinestyle erhält man für die Constants-Bezeichnung xlContinuous den Wert 1. Somit ergibt sich für die direkten COM-Steuerung aus APL folgende Syntax:

```
COM 'PROPERTY'  H¯EXCEL 'Ealection.
Borders().Linestyle'  7 1
```

Nach diesem Prinzip kann die gesamte Zielfunktion umgeschrieben werden, so dass sie sich im Endeffekt wie folgt darstellt (siehe Abb. 9):

```
[13]  →0×⍳↑(RC H¯EXCEL H¯BOOK)←1(3⊃ΔL)0 EXCEL_INI ''
[14]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'Application.ScreenUpdating' 0
[15]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'CELLS().RESIZE().VALUE'(1 1)(⍴ΔR)(ΔR)
[16]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'COLUMNS().SELECT'(⊂'A:I')
[17]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'COLUMNS.AUTOFIT'
[18]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'RANGE().SELECT'(⊂'A1')
[19]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.FONT.NAME'('Sparkasse Rg')
[20]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.FONT.SIZE' 14
[21]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.FONT.BOLD' 1
[22]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'RANGE().SELECT'(⊂'A2:A4')
[23]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.FONT.ITALIC' 1
[24]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'RANGE().SELECT'(⊂'A6:I40')
[25]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 7 1
[26]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 7 4
[27]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 8 1
[28]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 8 4
[29]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 9 1
[30]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 9 4
[31]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 10 1
[32]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 10 4
[33]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 11 1
[34]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 11 2
[35]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'RANGE().SELECT'(⊂'A6:I6')
[36]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 7 1
[37]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 7 ¯4138
[38]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 8 1
[39]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 8 ¯4138
[40]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 9 1
[41]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 9 ¯4138
[42]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 10 1
[43]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 10 ¯4138
[44]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().LINESTYLE' 11 1
[45]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.BORDERS().WEIGHT' 11 2
[46]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.INTERIOR.COLORINDEX' 3
[47]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.INTERIOR.PATTERN' 1
[48]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'RANGE().SELECT'(⊂'A7:I40')
[49]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.INTERIOR.COLORINDEX' 34
[50]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'SELECTION.INTERIOR.PATTERN' 1
[51]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'RANGE().SELECT'(⊂'A1')
[52]  →L_E×⍳1⊃1⊃1 COM 'PROPERTY' H¯EXCEL 'Application.ScreenUpdating' 1
[53]
[54] L_E:
[55]  0 0⍴1 COM 'METHOD' H¯BOOK 'Close'
[56]  0 0⍴1 COM 'METHOD' H¯EXCEL 'Quit'
[57]
[58] L_REL:
[59]  COM_RELEASE 'H¯BOOK' 'H¯EXCEL' 'RC' 'H¯VBC' 'H¯COMP' 'H¯CMOD'
```

Abb. 9: Direkte COM-Syntax (Ausschnitt)

Selbstverständlich kann man die Funktion an dieser Stelle noch um dynamisierte Zellbereiche verfeinern, die Demonstration soll aber hier beendet werden.

Fazit

Man kann bei der Konvertierung der VBA-Syntax in die direkte COM-Syntax leider keine Faustformel angeben. In Kenntnis der von mir angegebenen Hilfsmittel, angereichert um einen detektivischen Einsatz des Internets, lässt sich aber ein Wissen aufbauen, dass die direkte COM-Programmierung zu einem hervorragenden Werkzeug für die tägliche Arbeit mit dem IBM-APL2 macht.

Kontakt:

Frank Schmidt-Kelletat
Rheinischer Sparkassen- und Giroverband (RSGV)
Kirchfeldstraße 60, D-40217 Düsseldorf
Telefon: +49 (0)2 11 38 92-252
Telefax: +49 (0)2 11 38 92-323
eMail: frank.schmidt-kelletat@rsgv.de
Internet: Web: http://www.rsgv.de

APL (Programmiersprache)

APL, abgekürzt für **A P***rogramming* **L***anguage*, ist eine üblicherweise interpretierte Programmiersprache, deren Syntax überwiegend aus Symbolen besteht. Sie wurde von Kenneth E. Iverson und seinen Kollegen[1] bei IBM in den 1960er Jahren als algorithmische Notation (u. a. für den Mathematikunterricht)[2] entwickelt. Ab 1963 wurde APL als Programmiersprache auf Großrechnern (z. B. IBM System/360) implementiert und den Kunden als Timesharing-Option angeboten. Zu jener Zeit wurden IBM-Kugelkopf-Schreibmaschinen als „Terminal" verwendet. Daher hatte der ursprüngliche Zeichensatz nur (kursive) Großbuchstaben.

IBM 5100 Mikrocomputer mit APL-Tastatur

Ab 1975 gab es eine Implementierung auf der IBM 5100, einem Vorläufer des PC, der bereits ein Mikrocomputer war. Obwohl die Maschine nach damaligen Maßstäben als durchaus leistungsfähig galt, lief APL auf ihr verhältnismäßig langsam, da es sich um eine Emulation des Mainframe-APL-Interpreters handelte.

Die ersten Implementierungen auf PCs benötigten ein Spezial-EPROM, damit der APL-Zeichensatz angezeigt werden konnte. Der APL-Zeichensatz wurde ursprünglich so gewählt, dass die Symbole möglichst selbsterklärend sind, dass sich also die Funktionen aus den Symbolen intuitiv herleiten lassen. Seit einigen Jahren wird der Begriff APL auch als **A***rray* **P***rocessing* **L***anguage* interpretiert.

→ http://de.wikipedia.org/wiki/APL_%28Programmiersprache%29

Ralf Herminghaus

APL-Benchmark: Wie schnell ist APL wirklich?

Ursprünglich aus einer Weinlaune heraus entstand die Idee, die Geschwindigkeiten unterschiedlicher APL-Interpreter miteinander zu vergleichen und eventuell sogar den schnellsten und besten der auf dem Markt angebotenen Interpreter herauszufinden. Dieses Ziel ließ sich nicht erreichen. Die Beschäftigung mit Benchmarks im Allgemeinen und den Einzelergebnissen der hier vorgelegten Untersuchung im Besonderen fördert trotzdem einige interessante und teilweise überraschende Erkenntnisse zutage.

Einleitende Definitionen

Um Geschwindigkeitsmessungen bei unterschiedlichen Interpretern durchführen zu können, benötigt man zunächst einige Standard-Datenobjekte. Definiert sind daher im Folgenden einige Objekte wie numerische oder boolsche Vektoren, numerische Matrizen und auch einige Character-Objekte, die für die Messungen benötigt werden.

Die Datenobjekte

```
⍝ ⍝------------------------------------------------------------
⍝ ⍝ Test-Objects (N = General Size Parameter)
⍝ ⍝------------------------------------------------------------
⍝ X←(?Nρ1000)×0.001          ⍝---NUM-Vector--------------------
⍝ Y←(?Nρ1000)×0.001          ⍝---NUM-Vector--------------------
⍝ B←X>0.5                    ⍝---Boolean Vector----------------
⍝ IND_B←B/⍳⍴B                ⍝---Index-Representation of B ----
⍝
⍝ sN←10                                ⍝---Size of short Strings--
⍝ L←Nρ⎕AV[65+(?(N)ρ60)]                ⍝---Long String------------
⍝ S←sNρ⎕AV[65+(?(sN)ρ60)]              ⍝---Short String-----------
⍝ K←(N sN)ρ⎕AV[65+(?(N×sN)ρ60)]        ⍝---Character-Key-Table----
⍝ KX←⊂[2]K                             ⍝---Character-Key-Vector---
⍝
⍝ AV_FIRST←⍎⎕AV                        ⍝---First Element of ⎕AV---
⍝ AV_LAST←⍎¯1↑⎕AV                      ⍝---Last Element of ⎕AV----

⍝ sN2←20                               ⍝---Size of short Strings--
⍝ S2←sN2ρ⎕AV[65+(?(sN2)ρ60)]           ⍝---Short String-----------
⍝ K2←(N sN2)ρ⎕AV[65+(?(N×sN2)ρ60)]     ⍝---Character-Key-Table----
⍝ KX2←⊂[2]K2                           ⍝---Character-Key-Vector---
⍝
⍝ M←X∘.+X                              ⍝---Num.Matrix-------------
⍝ MX←⊂[2]M                             ⍝---Vector of Num.Vectors--
⍝ MC←(N N)ρ⎕AV[65+(?(N×N)ρ60)]         ⍝---Character-Matrix-------
```

Da die Messung auf allen getesteten Interpretern gleichermaßen lauffähig sein sollte, konnte der Algorithmus zur Geschwindigkeitsmessung nur auf Funktionen aus dem APL-ANSI-Standard basieren. Entsprechend einfach ist der Mess-Mechanismus gestrickt.

Er gliedert sich grob in vier Einzelschrittte:

Messung / Schritt 1

(Warten, bis die Systemuhr eine volle Sekunde zeigt)

```
      TS1←⎕TS
      ⍝----------------------------------
      ⍝ 1) Wait a  second ...
      ⍝----------------------------------
      LOOP01:
          TS2←⎕TS
          →((6↑TS2)≠(6↑TS1))/END_LOOP01
          →LOOP01
      END_LOOP01:
```

Messung / Schritt 2

(Eine Sekunde lang immer dasselbe tun – so oft wie möglich)

```
      ⍝----------------------------------------
      ⍝ 2) Measuring (for 1 Sec)
      ⍝----------------------------------------
      I←0
      LOOP02:
          I←I+1
          DUMMY←⍎'OPERATION'    ⍝---Do Operation---
          TS3←⎕TS
          →(I>I)/END_LOOP02   ⍝---Dummy-Compare--
          →((6↑TS3)≠(6↑TS2))/END_LOOP02
          →LOOP02
      END_LOOP02:
```

Messung / Schritt 3

(Genau so oft fast nichts tun)

```
      ⍝------------------------------------------
      ⍝ 3) Measuring Loop-Overhead
      ⍝------------------------------------------
      J←0
      LOOP03:
          J←J+1
          DUMMY←⍎'0'          ⍝---Do almost nothing---
          TS4←⎕TS
          →(J≥I)/END_LOOP03   ⍝---Non-Dummy-Compare---
          →((6↑TS4)≠(6↑TS3))/END_LOOP03
          →LOOP03
      END_LOOP03:
```

Messung / Schritt 4

(Am Schluß die gemessenen Timestamps auswerten)

```
      ⍝-----------------------------------------------------------
      ⍝ 4) Calculating Result
      ⍝-----------------------------------------------------------
      PATTERN←(1 1 1 1 60 60 1000)              ⍝ Timestamp-Structure
      USED_TIME←0.001×PATTERN ⊥ (TS3-TS2)       ⍝ (seconds)
      LOOP_OVERHEAD←0.001×PATTERN ⊥ (TS4-TS3)   ⍝ (seconds)

      TIME←1000000×(USED_TIME-LOOP_OVERHEAD)÷I  ⍝ micro-seconds
      RESULT←⌊(0.5)+TIME                        ⍝ Rounding
```

Verfahrens-Schwächen

Da bei dem hier vorgestellten Verfahren tatsächlich nur die absolut benötigten Laufzeiten gemessen werden, könnte der Test unter bestimmten Umständen irreführende Ergebnisse liefern. Um eine Fehlinterpretation der Messergebnisse zu vermeiden, soll hier daher kurz auf die Schwächen des Verfahrens eingegangen werden.

► MEMO-Mechanismen

Eine quasi funktionale Programmiersprache wie APL bietet dem Interpreter-Konstrukteur immer die Möglichkeit, Laufzeitverbesserungen über den Einsatz von MEMO-Mechanismen zu erreichen. Ein solcher Mechanismus greift immer dann, wenn eine Funktion mehrfach mit den gleichen Argumenten aufgerufen wird. Je nach Problemstellung lassen sich in der Praxis durch den Einsatz von MEMO-Mechanismen eklatante Geschwindigkeitsverbesserungen erzielen. Im vorliegenden Programm (Schritt 2) etwa würde die untersuchte Operation lediglich einmal tatsächlich ausgeführt. Für die restlichen Schleifen-Durchläufe würde der Interpreter dann lediglich auf das abgespeicherte Ergebnis zurückgreifen. Die Benchmark-Messung geht grundsätzlich davon aus, daß ein solcher Mechanismus im untersuchten Interpreter nicht implementiert ist.

► Lazy Evaluation

Ebenfalls aus dem Bereich der funktionalen Sprachen stammt die Strategie der "Lazy Evaluation". Die Operation wird hierbei in der Regel nicht sofort ausgeführt, sondern als nicht evaluierter Ausdruck abgespeichert. Erst wenn das Ergebnis der Operation tatsächlich benötigt wird, wird die Berechnug dann „nachgeholt" . Die Sprache HASKELL etwa benutzt diese Strategie. In den mir bekannten APL-Interpretern ist die Zuweisung einer Variablen auf eine andere in der Regel „Lazy" implementiert. Ansonsten geht der Test aber davon aus, daß ein solcher „Lazy Evaluation"-Mechanismus nicht vorliegt.

► Intelligente Datenfluß-Analyse

Schaut man sich die Schleife Schritt 2 einmal genauer an, so wird ziemlich schnell klar, daß das berechnete Ergebnis in der Variablen „Dummy" später nie wieder benötigt wird. Ein intelligenter Optimizer, wie er in Compiler-Sprachen wie C++ oder FORTRAN durchaus üblich ist, könnte daher die gesamte Schleife einfach „wegoptimieren".

► Garbage-Collector-Aufwände

Wann der Garbage-Collector eines Interpreters gestartet wird, entzieht sich in der Regel der Steuerung durch den Programmierer. Da je nach Implementation die Arbeit des Interpreters während des Garbage-Collector-Laufes unterbrochen wird, würde dieser Effekt die Messergebnisse deutlich verfälschen. Bei den durchgeführten Messungen deutet aber nichts auf einen solchen Effekt hin.

► Schwankende Betriebssystem-Leistung

Normalerweise stellt sich dieses Problem nur bei Time-Sharing-Systemen wie etwa MVS. Da aber auch Windows mittlerweile grundsätzlich immer mehrere Tasks gleichzeitig abarbeitet, ist auch hier mit derartigen Störungen des Meß-Vorganges zu rechnen. Grund dafür könnten eventuell Virenscanner sein oder andere Prozesse, die ohne Wissen des Benutzers im Hintergrund Kapazitäten beanspruchen.

► Zufall

Die Test-Objekte des Bechmarks sind über einen Zufallszahlen-Generator definiert. Daher ist es durchaus möglich, daß zufällige Daten-Konstellationen das Meßergebnis in der einen oder anderen Weise beeinflussen.

Getestete Interpreter

- APL+WIN Rel.9.1
- APLX Version 5.0.5 / MicroAPL Ltd
- IBM APL2 Version 2.0 / Service Level 18
- Dyalog APL 13.0
- NARS2000 Version 0.0.2.23

Verwendete Rechner

- Intel(R) Pentium (R) 4 mit 3.00 GHz
- RAM: 982.252 KB
- Betriebssystem: Windows 2000 / SP 3
- Celeron(R) mit 2.60 GHz
- RAM: 240 MB
- Betriebssystem: Windows XP / SP1

Die meisten Interpreter konnten ohne Probleme auf dem Pentium/Windows 2000- System installiert werde. Beim

Beispiel 1

Zunächst wollen wir einmal das Zeit-Verhalten einiger Boolscher Operationen betrachten.

Spitzenreiter ist hier der Interpreter von IBM. Hier sind die Zeitaufwände nahezu unabhängig von der Länge des bearbeiteten Vektors. Offenbar wird dabei die Tatsache genutzt, daß für die Bestimmung des Ergebnisses nicht unbedingt immer der gesamte Vektor verarbeitet werden muß.

```
⍝  ----------------------------------------
⍝  IBM APL2 VERSION 2.0 SERVICE LEVEL 18
⍝  Boolean Operations (\)
⍝  ----------------------------------------
⍝       N ^\B v\B ⍲\B ⍱\B  +\B ~B
⍝    1000   4   4   4   4  214  2
⍝    2000   4   4   5   5  426  2
⍝    3000   5   5   5   5  637  2
⍝    4000   5   5   5   5  847  2
⍝    5000   6   5   6   6 1054  2
⍝    6000   6   6   6   6 1266  2
⍝    7000   6   7   6   7 1473  2
⍝    8000   7   7   7   7 1692  2
⍝    9000   7   7   7   7 1894  3
⍝   10000   7   7   8   8 2105  3
⍝  ----------------------------------------
```

Bei den Interpretern von APL+WIN und NARS2000 wird dagegen offensichtlich immer der gesamte Vektor abgearbeitet. Die Aufwände steigen ungefähr proportional mit der Vektor-Länge:

```
⍝  ----------------------------------
⍝  APL+WIN 9.1
⍝  Boolean Operations (\)
⍝  ----------------------------------
⍝       N ^\B v\B ⍲\B ⍱\B +\B ~B
⍝    1000  23  24  23  23  29  3
⍝    2000  41  41  42  43  50  4
⍝    3000  59  59  61  62  77  4
⍝    4000  78  80  79  80  96  4
⍝    5000  97  95  99  95 109  4
⍝    6000 115 115 117 115 150  5
⍝    7000 132 132 136 132 154  5
⍝    8000 151 151 151 150 176  5
⍝    9000 171 170 173 173 213  6
⍝   10000 193 191 196 196 255 11
⍝  ----------------------------------
```

```
⍝  ----------------------------------
⍝  NARS2000 Version 0.0.2.23
⍝  Boolean Operations (\)
⍝  ----------------------------------
⍝       N ^\B v\B ⍲\B ⍱\B  +\B ~B
⍝    1000  81  81  82  82  1196 47
⍝    2000  98  92  98  99  2357 46
⍝    3000 108 108 108 107  3466 53
⍝    4000 117 116 117 117  4667 52
⍝    5000 134 135 128 129  5817 53
⍝    6000 159 168 166 159  6766 53
⍝    7000 157 158 156 155  7943 53
⍝    8000 168 169 168 169  9056 53
⍝    9000 179 179 181 179 10200 59
⍝   10000 193 193 193 192 11267 59
⍝  ----------------------------------
```

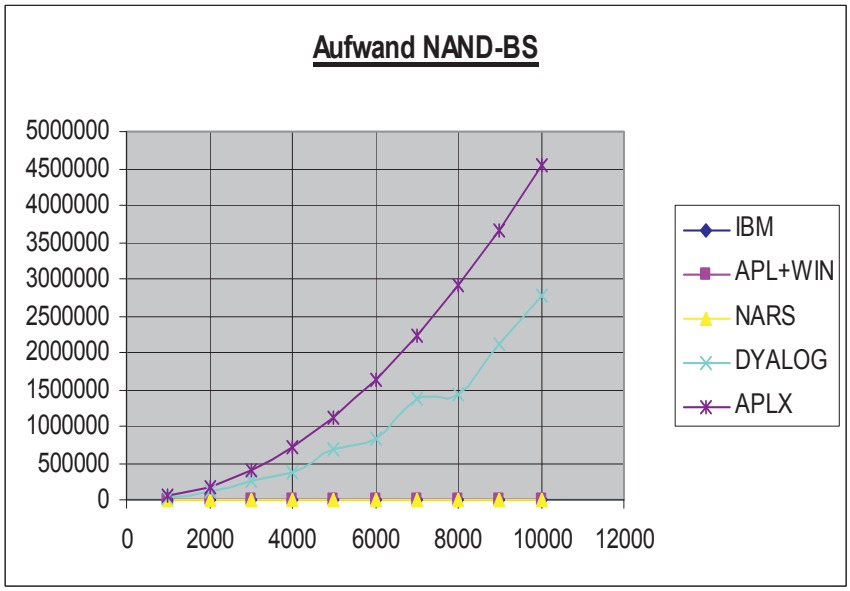
Auffällig bei den Interpretern von APLX und DYALOG ist der quadratisch steigende Aufwand für die NAND- und NOR-Operationen. Dies läßt darauf schließen, daß der Algorithmus für NAND-Backslash und NOR-Backslash „naiv“ implementiert wurde, ohne die algebraischen Möglichkeiten zur Optimierung auszunutzen:

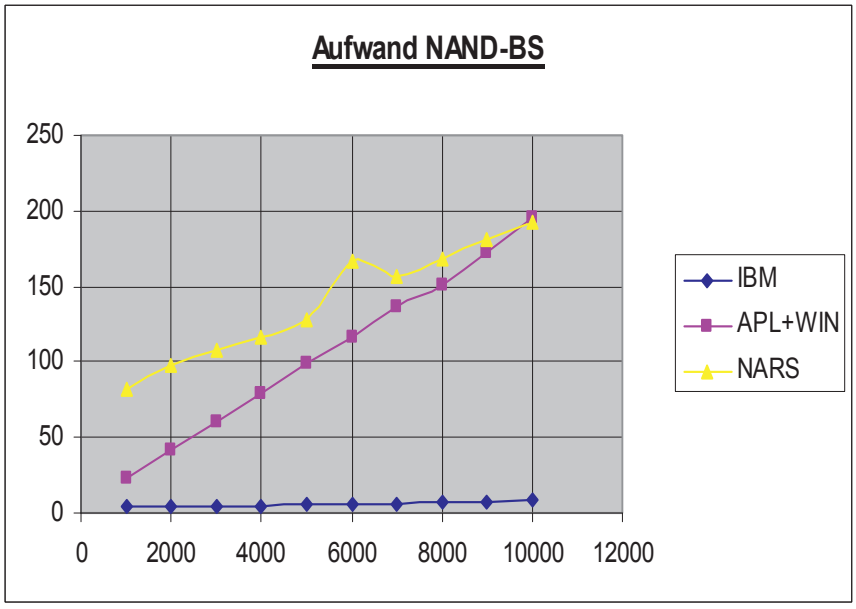
```
⍝  ----------------------------------------
⍝  APLX for Windows Version 5.0.5 / 7.0
⍝  Boolean Operations (\)
⍝  ----------------------------------------
⍝       N ^\B v\B      ⍲\B      ⍱\B +\B ~B
⍝  ----------------------------------------
⍝    1000  12  13    45455    44130  29  0
⍝    2000  24  24   184833   177000  58  0
⍝    3000  35  35   411333   395667  86  0
⍝    4000  47  46   726500   710500 116  0
⍝    5000  59  58  1125000  1109000 142  0
⍝    6000  70  70  1625000  1593000 177  0
⍝    7000  81  80  2218000  2156000 199  0
⍝    8000  94  93  2921000  2843000 230  0
⍝    9000  70 106  3671000  3593000 255  0
⍝   10000 117 116  4531000  4437000 287  0
⍝  ----------------------------------------
```

```
⍝  ----------------------------------------
⍝  DYALOG 13.0
⍝  Boolean Operations (\)
⍝  ----------------------------------------
⍝       N ^\B v\B      ⍲\B      ⍱\B +\B ~B
⍝    1000   0   0   27432   27778  40  0
⍝    2000   0   0  111111   90909  69  0
⍝    3000   0   0  250000  206200 165  0
⍝    4000   0   0  359333  364333 141  0
⍝    5000   0   0  687500  703000 275  0
⍝    6000   0   0  820000  820000 212  0
⍝    7000   0   0 1359000 1359000 392  0
⍝    8000   0   0 1421000 1453000 283  0
⍝    9000   0   0 2109000 1843000 313  0
⍝   10000   0   0 2765000 2765000 545  0
⍝  ----------------------------------------
```

Hier nochmals die fünf Interpreter im Vergleich:

NAND-BS					
N	IBM	APL+WIN	NARS	DYALOG	APLX
1000	4	23	82	27.432	45.455
2000	5	42	98	111.111	184.833
3000	5	61	108	250.000	411.333
4000	5	79	117	359.333	726.500
5000	6	99	128	687.500	1.125.000
6000	6	117	166	820.000	1.625.000
7000	6	136	156	1.359.000	2.218.000
8000	7	151	168	1.421.000	2.921.000
9000	7	173	181	2.109.000	3.671.000
10000	8	196	193	2.765.000	4.531.000

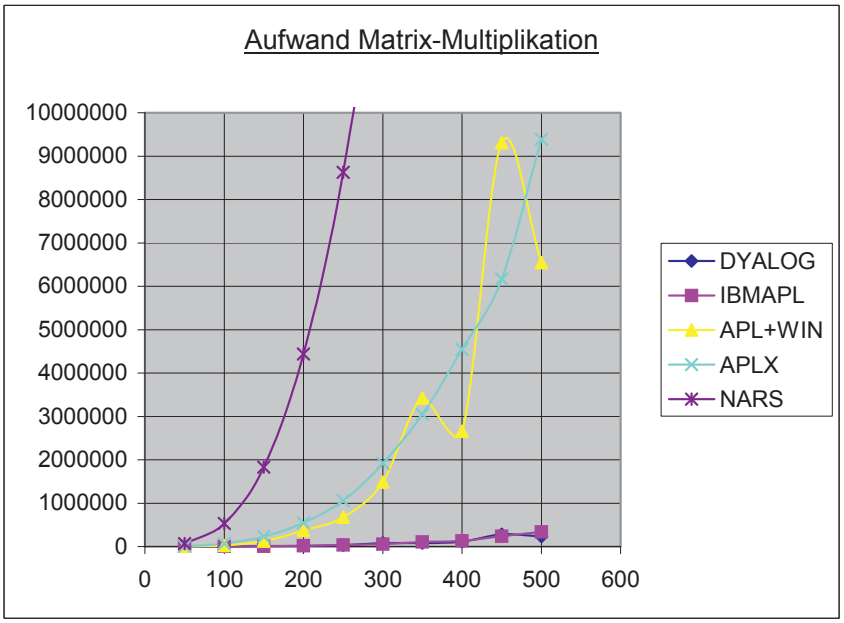




Beispiel 2

Technisch gesehen ist die Matrix-Multiplikation eine der wichtigsten Numerischen Operationen in der EDV überhaupt. Neben den klassischen Ingenieurs-Anwendungen wird sie in vielen Bereichen der Mathematik oder z.B. auch bei der Programmierung künstlicher neuronaler Netze benötigt. Entsprechend interessant ist die Frage, wie die verschiedenen Test-Kandidaten bei dieser Aufgabe abschneiden (siehe untenstehende Tabelle).

Meßergebnisse ihres Interpreters denn auch wesentlich aus einem Algorithmus, der die Leistungsfähigkeit des Cache-Speichers besonders gut auszunutzen vermag.



Matrix-Multiplikation (M+.×M)

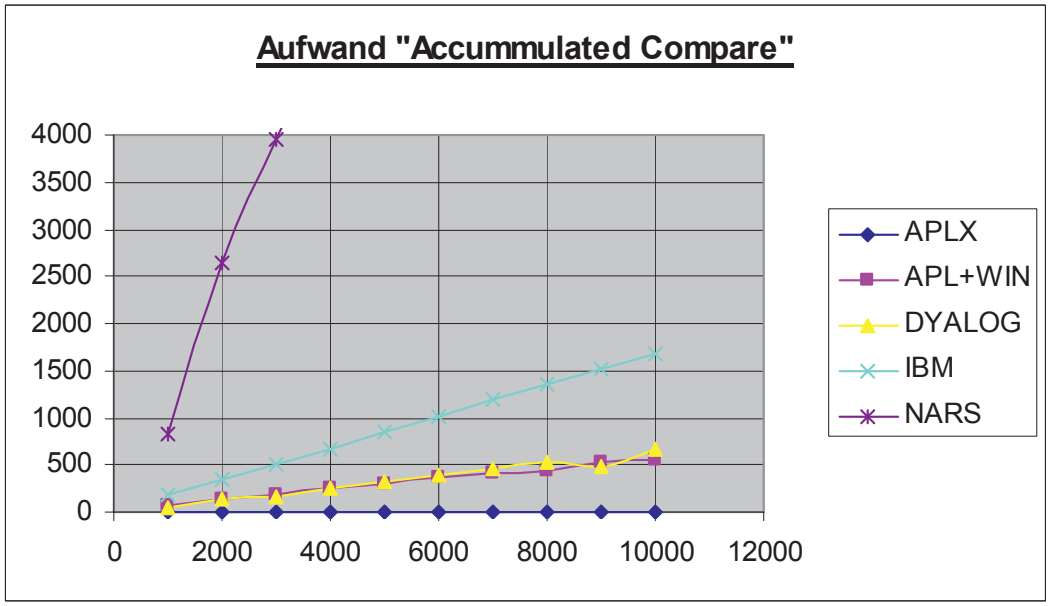
N	DYALOG	IBMAPL	APL+WIN	APLX	NARS
50	442	358	3356	8547	66667
100	2028	2717	27778	67667	531000
150	10989	7246	123222	234200	1828000
200	14706	17241	369667	546500	4437000
250	29000	40600	679500	1062000	8625000
300	83333	57278	1484000	1921000	14890000
350	84583	109300	3421000	3046000	24015000
400	112778	132750	2656000	4546000	35468000
450	281250	237400	9312000	6171000	51781000
500	234200	338333	6546000	9390000	70109000

Das gute Abschneiden der Interpreter von DYALOG und IBM ist dabei insofern bemerkenswert, als es prinzipiell nur schwer möglich ist, praktikable Matrixmutiplikationen zu implementieren, deren Aufwand geringer als kubisch ansteigt. Nach Angaben von DYALOG ergeben sich die guten

Beispiel 3

Das innere Produkt aus einem Vergleich „<“ mit nachfolgender Akkumulation durch AND ist insofern interessant, als sich hier eine offensichtliche Möglichkeit zur Optimierung bietet. Die Reihe der Vergleiche könnte nach dem ersten aufgetretenen „FALSE“ abgebrochen werden. Allerdings wird diese Möglichkeit offenbar nur vom APLX-Interpreter tatsächlich genutzt. Alle anderen Interpreter zeigen etwa linear steigende Aufwände, wobei trotz des prinzipiell gleichen Laufzeitverhaltens immer noch deutliche Leistungsunterschiede erkennbar sind.

Accumulated Compare (X^.<Y)					
N	APLX	APL+WIN	DYALOG	IBM	NARS
1000	5	65	52	174	823
2000	7	128	133	342	2636
3000	4	192	159	512	3958
4000	4	254	259	677	5126
5000	4	299	324	845	6446
6000	4	359	392	1022	7752
7000	5	419	454	1189	8972
8000	4	429	518	1353	10094
9000	4	531	486	1520	11588
10000	4	542	659	1684	12792

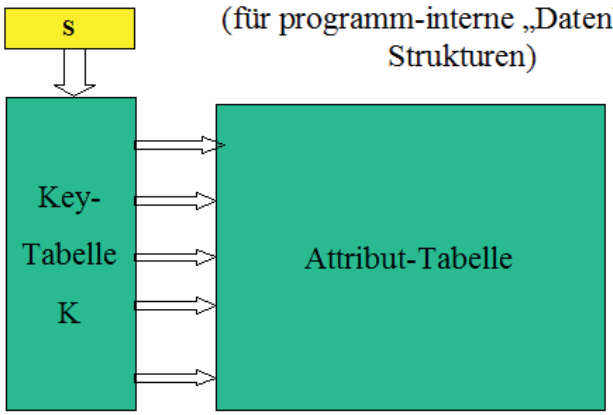


Beispiel 4

Die String-Suche in einer Tabelle von Schlüsseln ist vor allem deshalb interessant, weil sich hiermit innerhalb von APL-Programmen einfache datenbankähnliche Strukturen leicht implementieren lassen. Aus Effizienzgründen werden hierbei Primärschlüssel und Attribut-Tabelle in zwei unterschiedlichen Variablen abgelegt. Wesentlich für die Effizienz einer solchen „Datenbank“ ist dann die Geschwindigkeit, mit der Primärschlüssel lokalisiert werden können. Neben der klassischen String-Suche in einer Character-Matrix wurden auch Alternativ-Mechanismen für eine solche Funktionalität getestet.

String-Suche in einer Schlüssel-Tabelle

(für programm-interne „Datenbank“-Strukturen)



Auffällig ist dabei das ungewöhnliche Laufzeitverhalten des DYALOG- und NARS-Interpreters, das wahrscheinlich in Zusammenhang mit einer anderen internen Darstellung von Charactern erklärbar ist.

PART12: Key String search (length 10)						
N		APL+WIN	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	∨/S∈K	1.235	1.701	4.219	416	3.453
10000	(S∈K)[;1]	1.083	1.976	1.815	345	5.713
10000	K∧.=S	155	154	124	11.236	28.543
10000	KX⍳⊂S	1.464	311	20.300	2.427	7.067

(Erst im Nachhinein stellte sich heraus, daß die letzte hier aufgeführte Messung, KX⍳⊂S, für den DYALOG-Interpreter fehlerhaft ist)

Beispiel 5

Werden mathematisch äquivalente Ausdrücke auch tatsächlich äquivalent berechnet? Ein einfaches Beispiel liefert der Vergleich der Ausdrücke (-X) und (0-X). Dem intuitiven Verständnis nach sollten beide Ausdrücke etwa gleich aufwendig zu rechnen sein. Gegebenenfalls würde man erwarten, daß (-X) sogar etwas schneller berechnet wird, als der kompliziertere Ausdruck (0-X). Überraschenderweise zeigen die Interpreter von APL+WIN und DYALOG aber ein genau gegenteiliges Verhalten. (siehe Tabelle „Part1“). Die Ursache ist unklar.

Eine ähnliche Fragestellung ergibt sich bei mathematisch äquivalenten Zugriffen auf Elemente eines Arrays. Auch hier lassen sich die gemessenen Unterschiede nicht in jedem Fall plausibel erklären (siehe Tabelle unten „Part10“)

PART1: Standard-NUM-Functions						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	(-X)	289	90	152	485	802
10000	0-X	143	408	152	148	989
10000	÷X	480	236	192	4.831	1.011
10000	1÷X	472	542	196	4.444	948
10000	X*2	256	172	152	153	651
10000	X+X	194	178	152	144	1.439
10000	X*2	821	1.118	2.882	4.274	2.123
10000	X×X	202	236	151	142	1.269

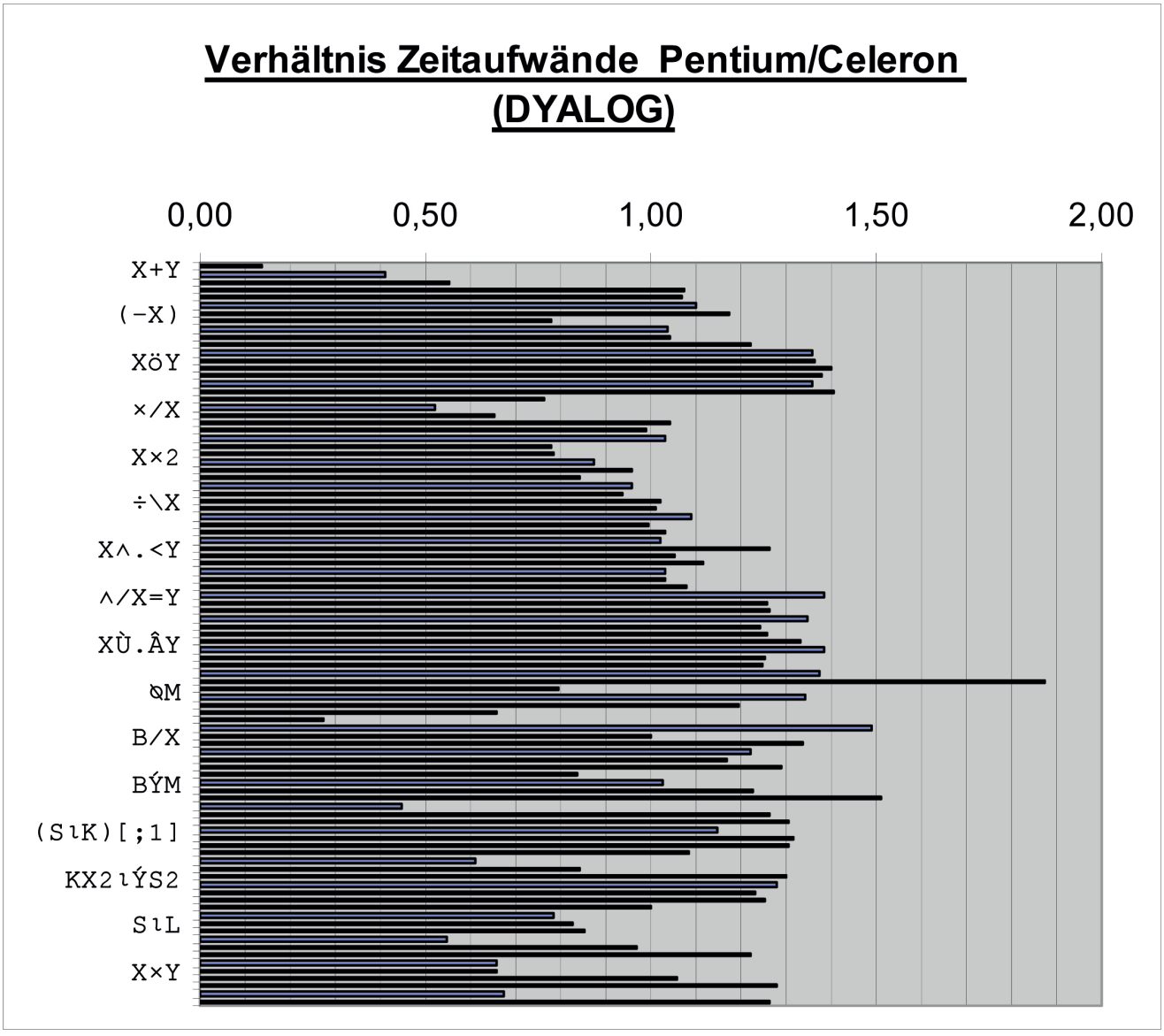
PART10: Selection						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
500	B/X	32	18	15	1	137
500	X[IND_B]	8	3	25	2	136
500	B/M	10.101	5.988	5.882	6.250	16.695
500	M[;IND_B]	2.770	9.615	10.638	2.890	56.389
500	B⌿M	1.876	3.289	3.257	2.193	33.300
500	M[IND_B;]	1.898	9.434	3.135	2.252	50.800

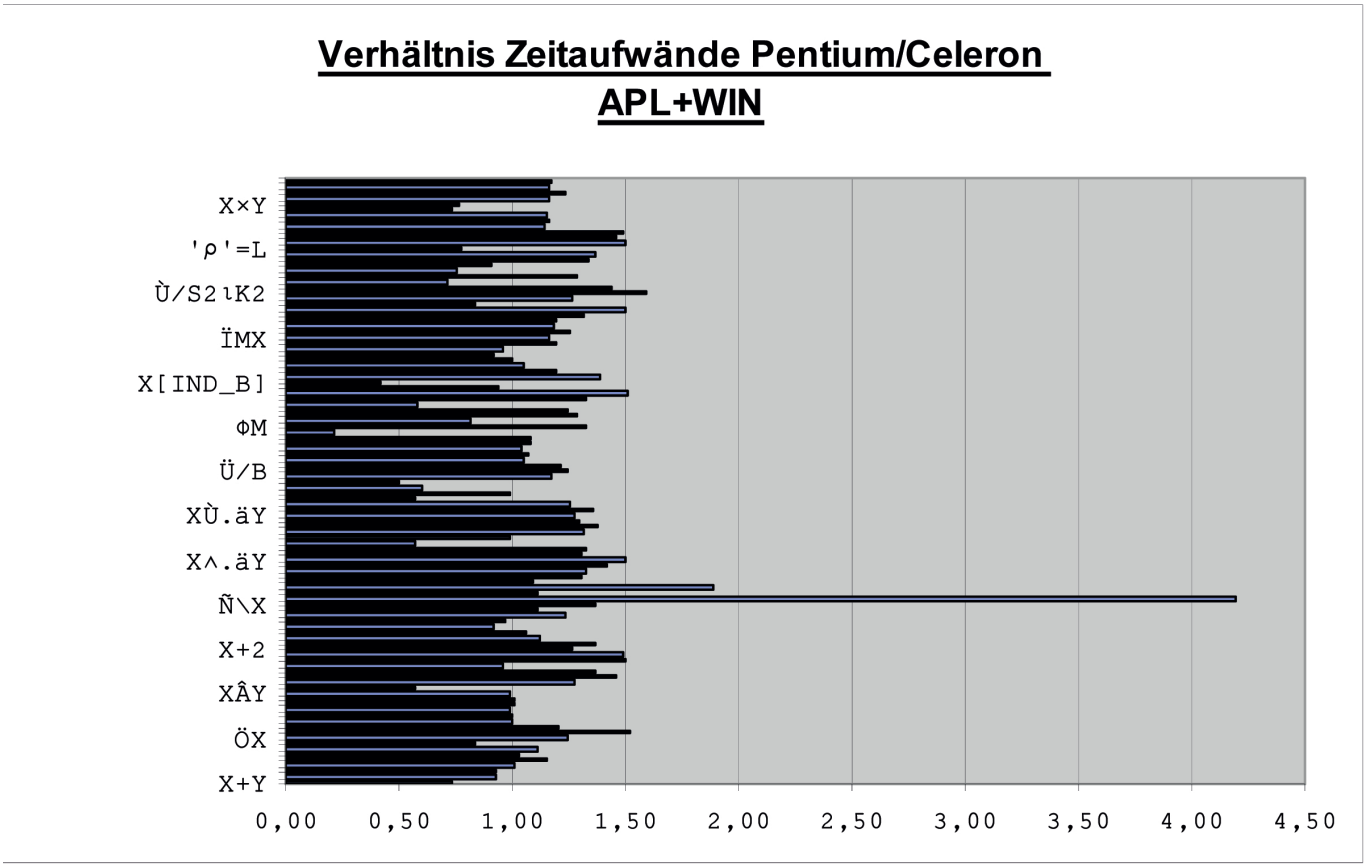
Wie sind die Meßergebnisse von der Hardware abhängig ?

Wie oben erwähnt, war es nicht möglich, alle Interpreter auf dem gleichen Rechner zu testen. Der NARS-Interpreter ließ sich auf dem benutzten Pentium-System nicht starten.

Um die Werte dennoch vergleichen zu können habe ich versucht, einen Proportionalitätsfaktor Pentium/Celeron zu bestimmen, der es ermöglichen sollte, die auf dem Celeron-System durchgeführten Tests auf das andere System umzurechnen. Wie die folgenden Meßergebnisse anschaulich zeigen, ließ sich ein solcher Proportionalitätsfaktor nicht ermitteln.

Dargestellt ist hier für alle durchgeführten Messungen das Verhältnis zwischen Pentium und Celeron, sowohl für den DYALOG- als auch den APL2000-Interpreter. Dabei zeigt sich deutlich, daß einzelne Operationen in Abhängigkeit vom Prozessortyp deutlich schneller oder langsamer ablaufen als andere. Eine Vergleichbarkeit der Prozessoren, wie man sie aufgrund von Taktfrequenz und Entwicklungsstand eigentlich erwartet hätte, ist jedenfalls nicht gegeben.





Fazit

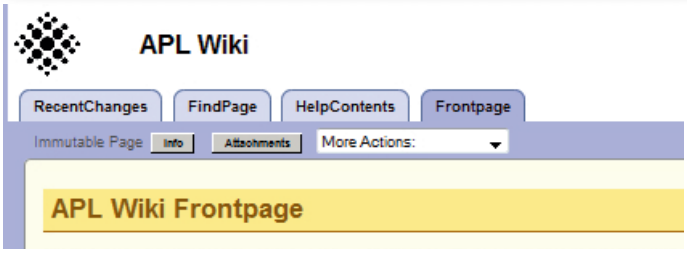
- Einen „Sieger“ oder „Verlierer“ des Benchmarks gibt es nicht.
- Die Meßergebnisse hängen von einer Vielzahl von Nebenbedingungen ab, die nur schwer zu kontrollieren und deren Einfluß kaum abzuschätzen sind.
- Die aus dem Benchmark abgeleiteten Aussagen sind daher alle mit einer sehr großen Unschärfe belastet.
- APL setzt man ohnehin nicht ein, um eine besonders hohe Performance zu erzielen.

Aber....

- Einzelne Ergebnisse sind durchaus sehr überraschend und lehrreich.

Und ....

- Für Verbesserungen ist sicherlich noch Potential vorhanden!



→ http://aplwiki.com/

Auflistung aller Meßergebnisse

PART1: Standard-NUM-Functions						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	X+Y	173	136	152	34	1.335
10000	X-Y	218	142	152	102	1.385
10000	X×Y	221	157	152	139	1.205
10000	X÷Y	338	244	199	4.545	1.392
10000	X*Y	4.049	2.404	2.865	6.452	2.820
10000	X!Y	13.158	289.000	13.889	33.833	28.571
10000	(-X)	289	90	152	485	802
10000	0-X	143	408	152	148	989
10000	÷X	480	236	192	4.831	1.011
10000	1÷X	472	542	196	4.444	948
10000	×X	215	128	149	387	1.279
PART2: Standard-Compare on NUM-Vectors						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	X<Y	606	471	427	554	5.330
10000	X≤Y	608	393	434	558	5.330
10000	X=Y	556	569	600	702	7.067
10000	X≥Y	607	370	425	555	5.330
10000	X>Y	605	475	436	553	5.330
10000	X≠Y	559	681	606	700	7.338
10000	X=Y	4	1	2	0	71
PART3: Accumulated Operations on NUM-Vectors (/)						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	+/X	182	209	36	91	1.911
10000	×/X	177	1.456	93	25	1.933
10000	⌊/X	97	159	110	39	1.777
10000	⌈/X	75	159	112	50	1.722
10000	-/X	215	201	52	860	1.924
10000	÷/X	355	314	243	5.155	1.968
10000	X+2	252	205	152	150	1.057
10000	X×2	256	172	152	153	651
10000	X+X	194	178	152	144	1.439
10000	X*2	821	1.118	2.882	4.274	2.123
10000	X×X	202	236	151	142	1.269

PART4: Accumulated Operations on NUM-Vectors (\)						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	+\X	224	2.740	157	522	11.011
10000	×\X	229	3.012	165	529	11.267
10000	L\X	302	2.058	164	530	10.533
10000	⌈\X	325	2.033	164	520	10.533
10000	-\X	2.031.000	10.578.000	2.320	2.515.000	50.203.000
10000	⍙MIN_BS	1.513	3.759	836	1.309	
10000	÷\X	1.796.000	129.453.000	1.171.000	22.843.000	55.968.000
10000	⍙DIV_BS	1.603	2.950	851	3.509	

PART5: Accumulated Compare of NUM-Vectors (∧)						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	X∧.<Y	761	4	1.689	662	13.873
10000	X∧.≤Y	809	4	1.695	551	13.873
10000	X∧.=Y	746	2	1.712	705	14.463
10000	X∧.≥Y	813	5	1.695	547	13.873
10000	X∧.>Y	764	4	1.669	545	13.873
10000	X∧.≠Y	701	1.013	1.745	703	12.423
10000	X≡Y	4	0	2	0	72
10000	∧/X=Y	559	570	600	704	7.178

PART6: Accumulated Compare of NUM-Vectors (∨)						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	X∨.<Y	799	5	1.767	662	13.458
10000	X∨.≤Y	839	4	1.786	666	13.458
10000	X∨.=Y	749	458	1.808	856	14.701
10000	X∨.≥Y	775	1	1.764	653	13.458
10000	X∨.>Y	734	4	1.779	663	13.458
10000	X∨.≠Y	703	3	1.835	857	12.628
10000	X=Y	4	1	2	0	71
10000	∨/X=Y	566	560	597	707	7.015

PART7: Boolean Operations (/)						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	∧/B	3	1	1	0	
10000	∨/B	3	0	2	0	
10000	⍲/B	189	918	183	924	
10000	⍱/B	189	893	186	923	
10000	+/B	113	71	7	0	

PART8: Boolean Operations (\)						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	∧\B	189	73	7	0	191
10000	∨\B	190	72	7	0	199
10000	⍲\B	194	3.812.000	7	2.765.000	190
10000	⍱\B	194	3.828.000	7	2.765.000	190
10000	+\B	204	287	2.049	545	11.867
10000	~B	11	1	3	0	57
PART9: Matrix Operations						
N		APL+WIN 9.1	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
500	⌽M	4.762	10.101	9.709	15.625	15.154
500	⊖M	12.987	41.042	10.309	6.369	50.000
500	⍉M	4.651	18.519	9.615	4.926	47.571
500	M+M	4.785	9.709	9.709	4.464	29.382
500	(M+.×M	5.875.000	7.859.000	411.333	390.333	51.109.000
500	(+/M)	4.566	5.208	845	3.906	44.826
500	(X∘.×Y	7.752	10.479	5.076	3.584	50.800

PART10: Selection						
N		APL+WIN	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
500	B/X	32	18	15	1	137
500	X[IND_B]	8	3	25	2	136
500	B/M	10.101	5.988	5.882	6.250	16.695
500	M[;IND_B]	2.770	9.615	10.638	2.890	56.389
500	B⌿M	1.876	3.289	3.257	2.193	33.300
500	M[IND_B;]	1.898	9.434	3.135	2.252	50.800
500	⊖B⌿⊖M	20.000	59.706	13.699	10.753	111.111
PART11: Restructuring						
N		APL+WIN	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
400	B⊂M	26.622	20.102	22.222	7.042	130.875
400	⊂[2]M	3.003	11.494	19.231	3.077	44.783
400	⊃MX	2.849	9.009	20.957	0	11.011
400	⌽M	2.933	6.494	3.802	8.000	9.500
400	⊖M	6.944	25.256	3.922	3.049	31.774
400	⍉M	2.801	10.638	3.690	2.950	23.452

PART12: Key String search (length 10)						
N		APL+WIN	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	∨/S∈K	1.235	1.701	4.219	416	3.453
10000	(S∈K)[;1]	1.083	1.976	1.815	345	5.713
10000	K∧.=S	155	154	124	11.236	28.543
10000	KX⍳⊂S	1.464	311	20.300	2.427	7.067
PART13: Key String search (length 20)						
N		APL+WIN	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	∨/S2∈K2	1.852	2.653	5.236	485	4.116
10000	(S2∈K2)[;1]	1.100	1.992	1.873	248	5.713
10000	K2∧.=S2	166	287	131	8.000	58.111
10000	KX2⍳⊂S2	1.468	301	26.026	2.457	7.067

PART14: Character search/String search						
N		APL+WIN	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	⎕AV⍳AV_FIRST	6	2	9	41	2.546
10000	⎕AV⍳AV_LAST	10	2	8	0	2.538
10000	(c⎕AV)⍳¨L	23.810	19.231	34.483	10.638	23.921.000
10000	(cL)⍳¨⎕AV	31.719	7.813	10.101	3.436	118.046.000
10000	'ρ'=L	67	78	37	44	235
10000	'ρ'∈L	535	25	17	144	494
10000	S∈L	509	7	31	87	525
10000	S2∈L	509	5	31	85	522
PART15: Polynoms						
N		APL+WIN	MicroAPL	IBM_APL	DYALOG	NARS(Cel)
10000	(0.2)⊥X	851	4.032	1.605	641	1.226
10000	(Nρ0.2)⊥X	894	4.115	1.751	879	2.182
10000	X⊥Y	869	4.098	1.595	907	1.959
10000	X+Y	174	157	151	145	1.304
10000	X×Y	182	231	151	144	1.126

Kontakt:
Dipl. Math. Ralf Herminghaus
AXA Konzern AG
Colonia-Allee 10-20; D-51067 Köln
Tel.: 0049 221 148 - 23279, Fax: 0049 221 148 - 44 23279
eMail: ralf.herminghaus@axa.de

Wolfgang Hengstler

Exploring the Origins of IBM DB2 Analytics Accelerator

Given the phenomenal response to the IBM DB2 Analytics Accelerator, the IBM Data Management Magazine is taking an in-depth look into the product and its origins. This is the second in a series of articles where we will look behind the curtain and peek into the products origins. We will be interviewing two Distinguished Engineers Namik Hrle, Chief Inventor of the offering and Guogen Zhang (Gene), lead DB2 innovator for the offering.

Namik, what were the driving needs you saw in the marketplace, that lead to the creation of the DB2 Analytics Accelerator?

Namik: In the last decade Business Analytics has significantly gained on importance and that trend is accelerating. Successful companies rely on Business Analytics as the main source of competitive advantages in the market. As they become mission critical the analytical workloads require the same reliability, availability and security as the transactional workloads. It is a common knowledge that System z with DB2 for z/OS provides these characteristics better than any other platform in the industry. System z also generates most of the transactional data which is the key input to Business Analytics. And still, in many cases the analytical workloads are not implemented on System z. There are several reasons for this. As I already implied, in the past these workloads did not require the highest quality of service provided by System z. The analytical workloads are also very resource (CPU, memory) intensive and the only way to execute them efficiently on general purpose databases like DB2 is to spend lots of time and effort in monitoring and tuning. In some cases not even the best skills and tools are enough to achieve appropriate performance. This led many customers to deploy specialized data warehouse databases despite the lower quality of services and complexity associated with moving the data, high levels of data redundancy, multiple platforms to manage etc. (Picture 1)

So, our objective was to create a solution that combines best of both worlds: DB2's industry leading reliability, availability, security and transactional engine, and a special-purpose analytical appliance (therefore: no tuning!) that delivers lightning performance for complex, data and processing intensive queries. I am convinced that the best answer to this challenge is a hybrid system that consists of two engines

Picture 1

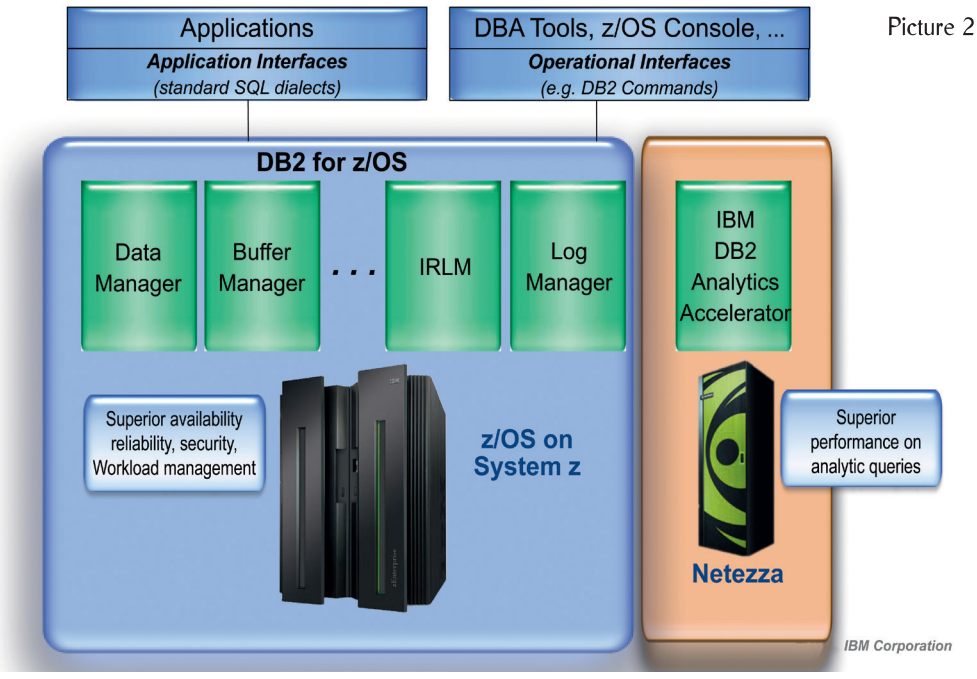
that are completely transparent to the users from both the programming and administration perspective. An appropriate analogy is a hybrid car. From the driver's perspective it is one, single car: one steering wheel, one accelerator pedal, one breaking pedal, ... but under the hood there are two engines that get activated transparently to the driver, based on various parameters that are monitored in real time.

How did Netezza become part of the solution?

Namik: My team started working on this idea a few years ago. At that time we decided to build our own analytical query engine, from ground up. So, we did. We took a prototype of an in-memory, columnar store query engine created by our Research division, and integrated it into DB2 in the hybrid fashion described above. The product became generally available in November 2010 as the first implementation of a hybrid database. Around the same time IBM acquired Netezza and we suddenly got opportunity to take advantage of the proven industry leading analytical appliance with unbeatable price/performance. We changed the train, so to speak, but stayed on the same tracks (**Picture 2**) leading into the same direction. This was the birth of DB2 Analytics Accelerator that brought the full power of Netezza to DB2 and elevated DB2 into a true universal database that incorporates two most powerful database engines in the industry.

What makes Netezza such a powerful query engine?

Namik: The Netezza® TwinFin® has a revolutionary design based on principles that have allowed Netezza® to provide the best



Picture 2

price/performance in the market. The four key components that make up TwinFin®: SMP hosts, snippet blades (called S-Blades), disk enclosures and a network fabric are carefully combined and preconfigured allowing the queries to be executed extremely fast. S-Blades are intelligent processing nodes that make up the turbocharged MPP engine of the appliance. Each S-Blade is an independent server that contains powerful multi-core CPUs, Netezza's unique multi-engine Field Programmable Gate Arrays (FPGA) and gigabytes of RAM; all balanced and working concurrently to deliver peak performance. FPGAs are commodity chips that are designed to process data streams at extremely fast rates. Netezza employs these chips to filter out extraneous data based on the SELECT and WHERE clauses in the SQL statement, as quickly as data can be streamed off the disk. The process of data filtering significantly reduces the amount of data, freeing up downstream components from unnecessary processing.

The S-Blades also execute an array of different database basic operations such as sorts, joins and aggregations in the CPU cores. The CPU cores are designed with ample headroom to run embedded algorithms of arbitrary complexity against large data streams for advanced analytics applications.

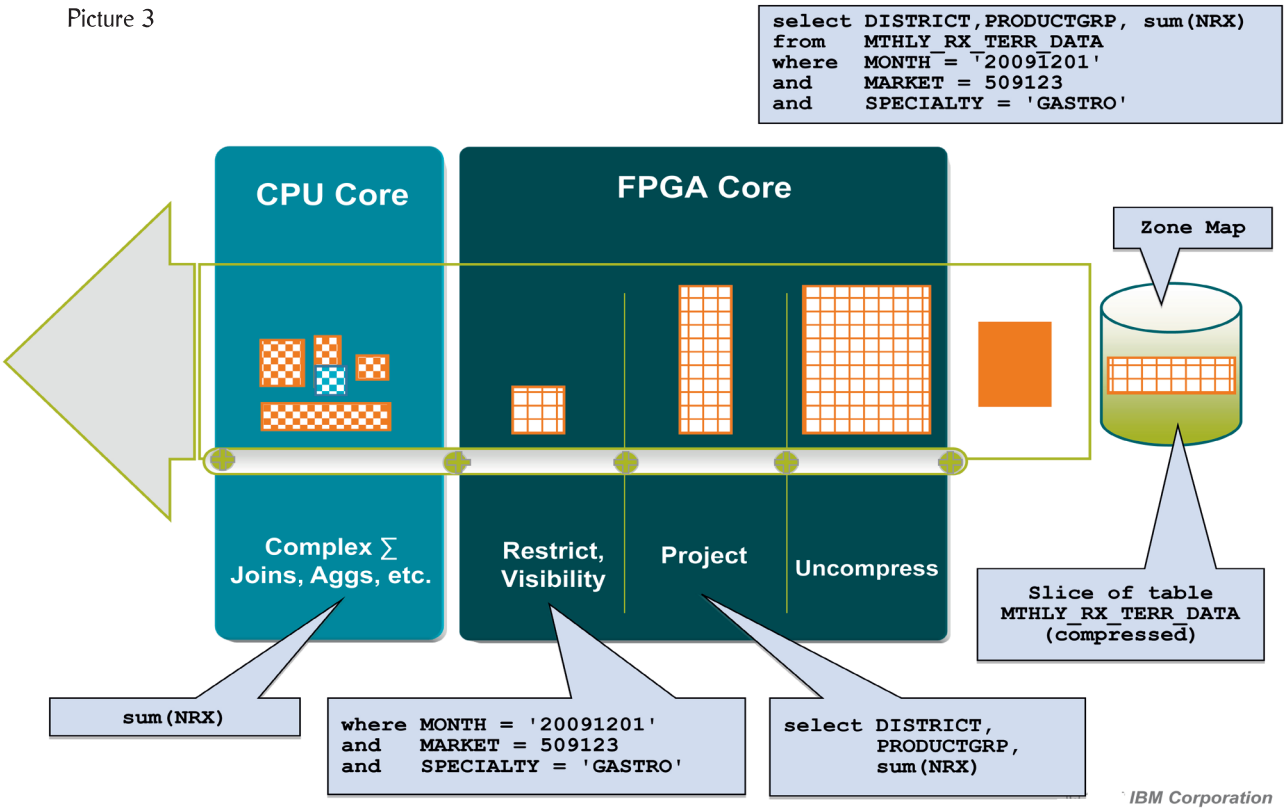
Gene, what is the basic concept, how Netezza technology is leveraged and integrated within the DB2 ANALYTICS ACCELERATOR product?

Gene: By leveraging Netezza appliance technology and integrating it with DB2 for z/OS®, we acquired a new query processing technology, almost instantaneously, with industry leading performance for complex analytic workloads at very affordable price point on a premium OLTP platform, i.e. System z, for our customers. Now our DB2 users can take advantage of this new technology to analyze vast transactional data on the same platform transparently to their competitive advantages. DB2 query processor analyzes queries and route queries typically seen in business intelligence (BI) and data warehousing applications from the System z environment to a workload-optimized accelerator, i.e. a Netezza server. The DB2 Analytics Accelerator complements DB2's traditional query processing, and can speed up a substantial percentage of queries, and it changes the equation in cost and performance, all transparently to the end user or applications. It is fast and easy to deploy, and it can be easily activated and deactivated with the setting or clearing off a software switch.

The key differentiation of DB2 Analytics Accelerator in relation to other offerings in the BI/DW space is its deep integration into DB2 which results in many advantages.

But before we go through that let me quickly describe the basic DB2 processing structure. DB2 consists of multiple internal components that we call Resource Managers: Buffer Manager for data cache management, IRLM for locking etc. These components are not visible to the outside world of applications and database administration which connect to DB2 through the application and operational interfaces such as SQL and commands or utilities etc. The internal processing flow among

Picture 3



the components is completely transparent to the outside world. So, if we add another Resource Manager into DB2 and adhere to the existing interfaces, no application or database administration procedure need to change. That's exactly the design decision that we made with DB2 Analytics Accelerator. (**Picture 3**)

For the outside world, the DB2 Analytics Accelerator is an integral part of DB2, transparent to DB2 users, just like all other DB2 components that are behind standard DB2 APIs. Having the query accelerator within DB2 reduces the need for moving data outside of the Enterprise Warehouse database, which is a common practice when creating data marts to accelerate specific segments of the database.

But it is not only about data and movement of data. It is also about optimization where to run a particular type of query workload. This DB2 Analytics Accelerator solution is able to optimize a mixed query workload automatically, and to execute queries in the most efficient way.

So DB2 makes query acceleration decision based upon the query characteristics, right?

Gene: Exactly. If the two technologies were not tightly integrated then developers or users would have to try to figure which query to submit to which components, DB2 or the accelerator. With the deep integration, it's not something that the end user has to worry about. The DB2 query processor analyzes each query and determines the fastest and most cost effective way to execute each query. There is no need for manual intervention of any kind. Query processing using the accelerator is like a new access path, similar to an index, or MQTs.

Once you have it installed and have data loaded into the accelerator, DB2 will utilize it for the queries, the best way it can. Most organizations have a database analyst who spends a great deal of time looking at query plans for their warehousing systems, trying to figure out whether to add an index to a table or rewrite the query. All of this inspection and work goes away. The SQL is moved out to the hardware accelerator so the end user and the database administrator don't have to worry about it. You continue to use the same applications, and they have much faster query response time. And the accelerator frees up more z CPU for transactional processing. The data retains its system-of-record in DB2 for z/OS and retains all of the security and operational attributes of DB2 data. The Accelerator is fully managed via DB2 stored procedures, making it a truly plug-and-play appliance.

With its processing power on the vast data in DB2 for z/OS, the customers can now run queries that were forbidden and forgotten due to their high costs or overly long response time, and create more analytic workloads to satisfy business needs. The potential is endless.

So Namik, what have been the first customer experiences?

Namik: Many people today talk about appliances, but this offering truly meets the definition. Our customers are typically able to run queries in a couple of days after the

No	Description	Records read	Records returned	DB2 [sec]	IDAA [sec]	Accel. factor
1	Simple mass aggregation	17116647	21	117	0.78	150
2	Query #1 + 70% filter	11980812	21	94.2	0.86	110
3	Query #1 + 30% filter	5133708	21	54.8	0.82	67
4	Query #1 + 10% filter	1710293	21	17.6	0.87	20
5	Screwed data, low filtering	10790019	21	96.8	2.47	39
6	Screwed data, high filtering	24	14	7.28	0.83	9
7	Many restrictions	3805941	21	128	7.65	17
8	Navigational attributes	823646	21	17.1	1.27	13
9	Navigational attributes + selective condition	811	21	15.8	1.17	14
10	Open value ranges	2006	21	19.6	3.52	6
11	Hierarchy	1653981	21	17.6	0.97	18
12	Hierarchy + selective condition	55068	21	38.6	0.98	39
13	Restricted key figures on 2 dimensions	1314964	1948	207	7.22	29
14	Query #14 + hierarchy	132564	1499	> 1000	1.27	> 787
15	Calculated key figures (OLAP)	5321586	10	57.8	2.37	24
16	OR linked values	6212609	13	40.5	0.92	44
17	Non uniform data distribution	11016253	13	31.2	0.99	32
18	Selective line item	1724	1706	0.71	1.17	0.6
19	Non-selective line item	115481	68619	33.8	1.36	25
20	All together	3087692	468	87.7	4.42	20

Picture 4

box arrives. There is no need for schema changes, for application adjustments, for data conversion. You attach IBM DB2 Analytics Accelerator DAA to DB2, specify tables to be accelerated, prime them in IDAA and your queries can take full advantage of the special purpose data warehouse appliance.

How does this change the dynamics of the architecture and why is that so important to customers?

Namik: For some data warehouse applications, the data has often been moved to other platforms to provide higher speed analysis. With the new DB2 Analytics Accelerator there is no reason to do that and pay the price in more complex landscape, separate system and administration tasks etc. In fact, DB2 now provides a platform that enables organizations to blend both their operational data store (ODS) and their data warehouse. In a typical data warehouse environment, data is extracted from the operational systems, put through a transformation and loaded into the data warehouse. That movement increases data latency. It also creates a lot of cost and overhead in moving that data around, opening the data to possible intrusions. With System z, everything is kept on the same platform, and DB2 and SQL is used to move data and do the transformations; the data never leaves the platform. This provides the ability to lower data latency, reduce costs, and make it a simpler overall environment. (Picture 4)

The product has been well received by the users community and we have a significant number of sales and many more customers who are strongly considering how they utilize this game changing technology.

Thanks Namik and thanks Gene for this interesting insight.

Autor und Interviewer:

Wolfgang Hengstler
Produktmanager DB2 Analytics Accelerator
IBM Deutschland Research & Development GmbH
Executive Briefing Center
Schönaicher Straße 220, D-71032 Böblingen
e-Mail: HENGST@de.ibm.com

Interviewpartner:
Namik Hrle
IBM Software Group
Distinguished Engineer

Guogen Zhang (Gene)
IBM Software Group
Distinguished Engineer

Peter Thaben

Warum eigentlich Linux?

Wer benutzt eigentlich Linux auf seinem Desktop? Was sind das für Menschen, die sich aus Sturheit, eitlem Elitedenken ohne das nötige Kleingeld – sonst würden sie wahrscheinlich auf einen Mac starren – oder gar aus Prinzip dem „eigentlichen und wahren“ Betriebssystem Windows verweigern? Hin und wieder tauchen im Zeitschriftenregal Extranummern von c‘t oder PC-Welt zum Thema Linux auf. Ganz hinten im Regal, eingequetscht zwischen den Gamer- und Photomagazinen stehen die Heftchen für das Hardcorepublikum: Linux User, Easy Linux und das Linux Magazin. Die ganze Welt ist vom Betriebssystem Windows besetzt … Die ganze Welt? Nein! Eine kleine unbeugsame Gemeinde von Mainstreamverweigerern frönt einer exotischen Leidenschaft: Sie benutzt Linux und — man mag es kaum laut sagen — sie ist glücklich damit und vermisst nichts. Ok, fast nichts. Aber dazu kommen wir gleich.

Dem normalen Benutzer eines PC oder Notebooks, der sich nicht um solche Feinheiten wie der verbauten Grafikkarte, dem Audiosystem oder gar dem Betriebssystem seines Rechners kümmern will, ist es wahrscheinlich völlig unbekannt, dass es Computer gibt, die nach dem Starten keinen Klingklang-Ton von sich geben und den erfreuten Benutzer nicht mit einer grünen Wiese unter blauem Himmel begrüßen. Wahrscheinlich wissen viele gar nicht, dass sie mit Windows arbeiten. Und sie müssen es ja auch nicht wissen. Ist ja alles schön vorinstalliert beim Kaufen. Anmachen und wohl fühlen. Alles schon an Bord: Musik anhören, Videos gucken, Texte schreiben, im Internet surfen (halt! noch schnell einen Virenscanner installieren), Bilder speichern, Filme runter laden, toll. Rasch noch die nervige Reklame deinstallieren, die den Laden anpreist wo man die Kiste gekauft hat, die Büroprosoftware ist nur die kostenlose Sparversion und müsste eigentlich aktualisiert werden, und was ist das für ein hektisches Fenster, das einen ständig nach irgendwelchen Dingen fragt. Kann man hoffentlich irgendwie ausschalten. Aber sonst – Windows ist prima! Mehr braucht kein Mensch. Mehr hat auch fast kein Mensch.

Auf der Grafik zur Betriebssystemverbreitung (siehe Abbildung 1) ist ganz unten am Rand eine rote Linie („Linux“) — was machen die eigentlich?

Wenn Windows ein perfektes Betriebssystem darstellt, dann braucht es eigentlich kein weiteres System. Gut, die Kreativen und die Feingeister, die da meinen, dass der

richtige Computer vor allem eine Lebenseinstellung ist, die nehmen den Mac. In Ordnung, dafür zahlen sie auch ein wenig mehr und haben – ohne Neid – auch eine ordentliche Technik unter den Fingern. Dass sie eigentlich auch Linux benutzen, verraten wir jetzt noch nicht.

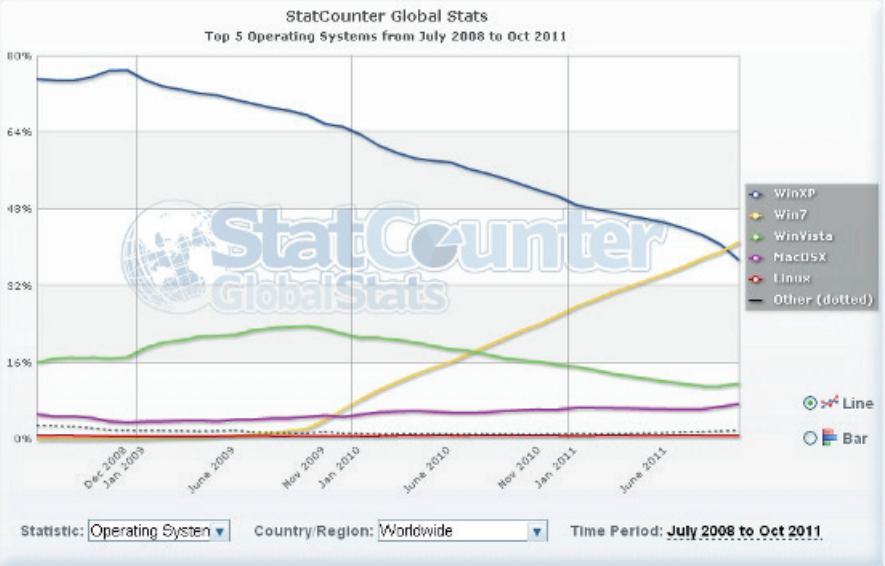


Bild 1: Betriebssystemverbreitung

Aber die kleine verkniffene Minderheit mit dem roten Balken, womit daddeln die denn herum?

Linuxer wird man nicht freiwillig

Keine Ahnung, wie die anderen zu Linux gekommen sind. Bei mir war der Arbeitgeber ein wenig daran Schuld. Bei der Berliner Stadtreinigung wurde Ende der 1980er Jahre für die Tourenplanung in der Straßenreinigung ein mehrplatzfähiges System gesucht. Die PC-Anwendungen mit dBase und Clipper waren nicht mehr ausreichend.

Auf den Großrechnern der BSR lief damals BS 2000 von Siemens und deren Beraten empfahlen uns SINIX-Maschinen. So kam ich zu UNIX und programmierte die Papierkorbentleerung und die Gullyreinigung fortan in 4GL.

Das war 1987. An einen Heim-PC war damals überhaupt nicht zu denken. Was sollte ich denn mit so einer Kiste zu Hause anstellen?

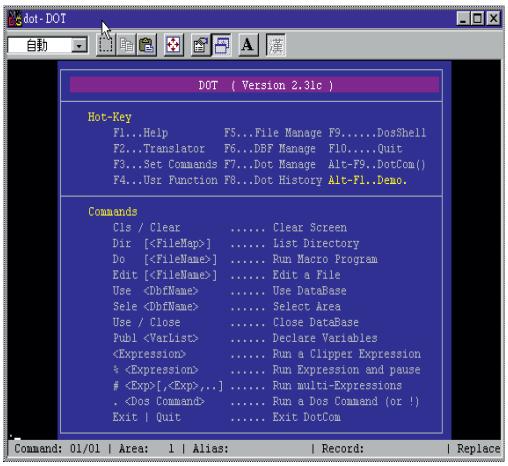


Bild 2: dBase

Den ersten privaten PC kaufte ich etwa zehn Jahre später. Ein Angebot aus einem Supermarkt mit dicken Bildschirm und Windows 95 als Betriebssystem. Damit konnte ich Texte schreiben, irgendwelche Spiele spielen und wenn ich das Telefonkabel einstöpselte, konnte ich mich über AOL ins Internet einwählen. Wie Boris Becker war ich „drin“. Das war damals sehr aufregend. Obwohl ich dienstlich täglich mit Computeranwendungen zu tun hatte, war mir das Funktionieren dieser grauen Kiste ziemlich unbekannt.

Bild 3: AOL

Ich bin heute kein großer Bastler und war es früher auch nicht. Das Ding sollte einfach funktionieren und wenn ich mal Lust hatte ein neues Programm auszuprobieren, dann machte ich das, was alle machten: herunterladen und installieren. Und es gab vieles herunterzuladen und zu probieren. Wenn es nichts war, konnte man es ja wieder löschen.

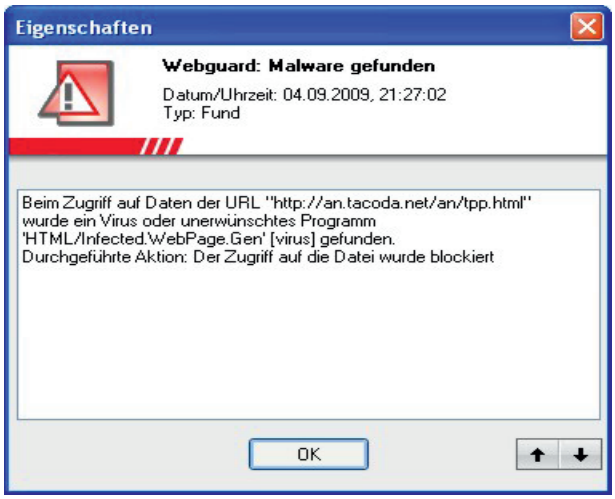


Bild 4: Viruswarnung

Meist klappte das ja auch ganz gut. Manchmal aber nicht. Dann blieb offenbar etwas zurück. Dann gingen plötzlich irgendwelche kleine Windows nicht mehr zu. Nicht mal ein Neustart half dagegen. Aber inzwischen machten kleine Helfer für den Internetnutzer von reden: Virenscanner. Und die fanden bei mir ständig was zum Meckern. Das war öde. Permanent irgendwas deinstallieren, ständig auf der Hut sein, nicht nur pünktlich die Updates und Bugfixes einspielen, sondern das Gerät auch noch regelmäßig „säubern“ und ordentlich „aufräumen“. Es muss doch etwas geben, was weniger aufregend ist. Linux vielleicht? Im Büro läuft es ja auch ganz sauber.

Mein erstes Versuchskaninchen war SuSE Linux, Version 7.2 im Jahr 2001. SuSE gab es in deutsch von der Gesellschaft für Software- und Systementwicklung GmbH auf einer Menge Installationsdisketten. Das war in den Zeiten eines noch etwas schwachbrüstigen Internets ganz hilfreich.

Das Ausprobieren von SuSE dauerte nicht lange, dann bin ich reumütig zu Windows zurück. Vieles war doch etwas frickelig. Vor allem fehlten Treiber für die Hardware und Programme waren auch eher knapp und nicht besonders gut. Also doch besser Windows – erst einmal. Es ist eben doch ein Unterschied zwischen einem Privatrechner und einem Bürosystem. Bis zur nächsten ernsthaften Annäherung an Linux vergingen einige Jahre.

Ist Windows nicht doch besser?

In dieser Zeit machte ich mich mit Windows und eigentlich noch mehr den verschiedenen Anwendungen vertraut, die man auf seinem privaten Computer eben braucht - oder mindestens meint zu brauchen. Und Anwendungen gab es wie Sand am Meer. Für alle Wechselfälle des

Lebens war da ein Programm: Texte bearbeiten und Bilder verwalten, hinterher alles drucken, PDF-Dateien lesen und editieren, Videos angucken und vielleicht sogar schneiden, Musik sammeln, sortieren und die Tags dazu aktualisieren, alles was man sich vorstellen kann, von dem Meer von Spielen gar nicht zu reden. Zu den richtigen Programmen kamen die vielen kleinen Helferlein, für den Download, um die Festplatte zu partitionieren, um die viele Musik auf CD brennen und ein passendes Cover dazu zu basteln, um Bildschirmschoner und Bildschirmhintergründe einzurichten, um den Rechner schneller, leiser und effektiver zu machen. Tausendundeine Idee, fast wie heute, wo es für alles, was man sich im einigermaßen wachem Zustand vorstellen kann, Apps gibt.

Die Programme waren aber keine Apps, die für einen einigermaßen begrenzten Nutzen und überschaubaren Nutzen verwendet werden. Programme nisteten sich in die Tiefen des Rechners ein, verstreuten auf verschlungenen Wegen ihre kleinen Bibliotheken und ihre Runtime-Utilities, irgendwo waren die Startparameter versteckt und ganz woanders die Prozedur, die sich bei jedem Start des Rechners reckte und - wichtig, wichtig - ihren Platz im knappen Hauptspeicher beanspruchte. Wer viele Programme hatte, konnte über eine kleine Armee solcher Startprozeduren befehlen. Wobei „befehlen“ eigentlich nicht der richtige Begriff ist. Unangenehm war dazu, dass sich irgendwelche Prozesse von längst wieder deinstallierten Programmen immer noch zur Arbeit aufgerufen fühlten und brav starteten ohne zu merken, dass sie im Weg herumstanden und den Platz versperrten.

Bild 5: Registry Cleaner

Das machte den Rechner irgendwann träge und stellte den User vor die Alternative: Neuinstallation oder neuen Rechner kaufen. Das war das eher harmlose Dilemma des Windows-Nutzers. Unangenehm wurde es wenn man sich bei seinen vielen Installationsversuchen im nächtlichen Downloadrausch was eingefangen hatte. Einen Wurm zum Beispiel, oder einen Virus. Meist merkte man das nicht gleich, sondern erst wenn was nicht wie gewohnt funktionierte. Irgendein Programm ließ sich nicht mehr richtig starten, merkwürdige Fenster ploppten plötzlich auf, tausende interessante Mails blockierten den Posteingang, oder es ging gar nichts mehr. Die Kiste fuhr gar nicht erst hoch. Diagnose: der PC ist infiziert, eine Wurmkur musste her. Die Damen und Herren Kaspersky, BitDefender, Norton oder McAfee hatten alle Hände voll zu tun, unsere heimischen Rechenroboter heil durch die Abgründe von Schadsoftware, Trojanern und Hackerangriffen zu lotsen. Das erforderte eine gewisse Disziplin und neben dem aktuellen Betriebssystem (Dienstag ist Patch-Day bei Windows) musste noch auf die neueste Version des Virenscanners geachtet werden.

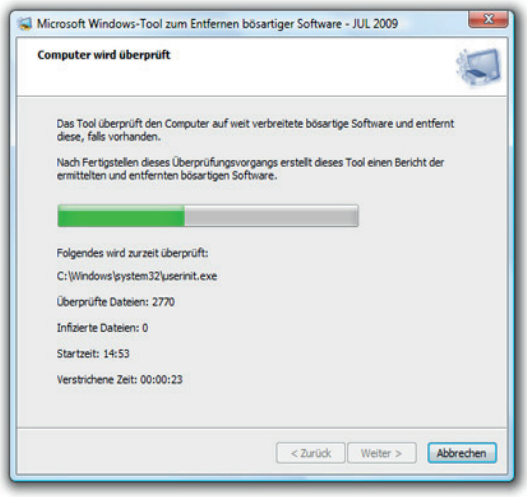


Bild 6: Windows Patch Day

Heute noch ist es die oberste Regel bei einer Windows-Neuinstallation: wenn das Betriebssystem drauf ist, zuerst das Antivirusprogramm starten und erst danach die Updates aus dem Netz zu holen. Uff!

Welches Linux hätten Sie denn gerne?

Also doch mal wieder Linux probieren. Die Entwicklung in der Nische der Unix-Derivate war inzwischen auch nicht stehen geblieben. Eine reichlich unübersichtliche Menge von originellen Distributionen mit witzigen Namen samt ihren unzähligen Neben- und Unterentwicklungen bietet sich dem neugierigen Laien, der einen Schritt vom breiten Pfad der Windowswelt hin zu den Pinguinen wagt.

Auf dem Bild 7 kann mal sich beispielsweise die Ableger des im Moment beliebtesten Betriebs-systems Ubuntu anschauen. Dabei ist Ubuntu nur eine von vielen Debian-Töchtern, die alle auch unzählige Kinder haben. Und neben Debian gibt es ja noch Red Hat, SuSE oder Mandriva als größere Entwicklungsquellen. Alle basieren auf dem quelloffenen UNIX-System. Und alle werden mehr oder weniger intensiv von mehr oder weniger Freiwilligen betreut, erweitert und regelmäßig in neuen Versionen dem geneigten Publikum angeboten.

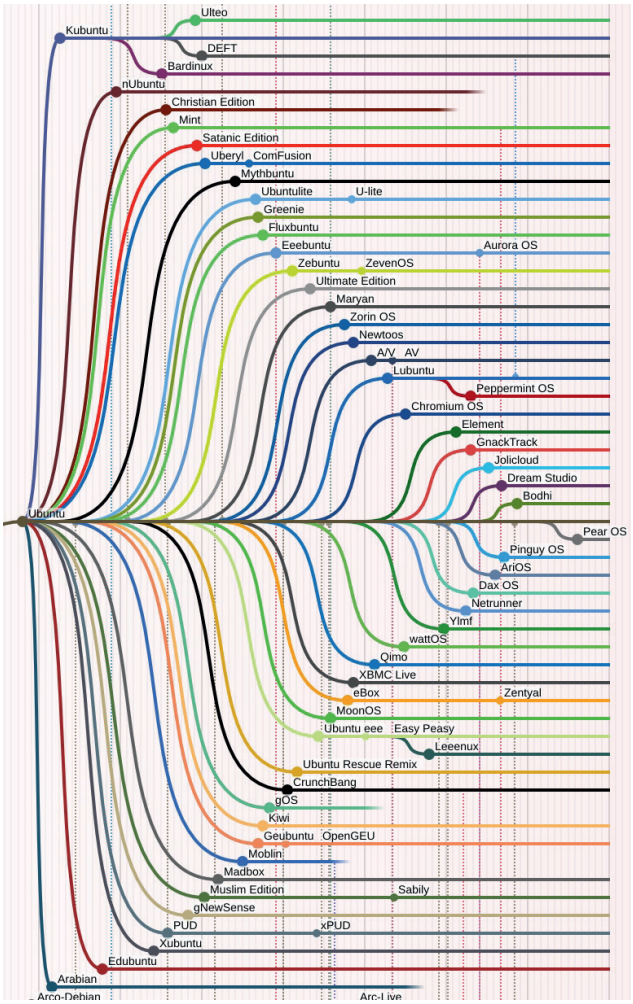


Bild 7: Linux Derivate

Bei dieser Unübersichtlichkeit sind die normalen PC-Benutzer zu recht überfordert. Da nimmt man doch lieber irgendwas mit Windows oder gleich einen Mac (Vorsicht, da steckt aber auch UNIX drin!).

Zum Glück gibt es auch bei Linux die Varianten für die Normalos, die einfach zu installieren, problemlos zu ergänzen und

dazu hübsch anzusehen sind. Doch, das ist auch wichtig.

In den vergangenen Jahren hat sich vor allem Ubuntu einen Namen als benutzerfreundliches System mit Pfiff gemacht. Mit seinen direkten Vettern Kubuntu und Xubuntu ist es inzwischen zu einer Marke geworden, die für „Linux auf dem Desktop“ steht.

Bild 8: Kubuntu Ubuntu Xubuntu (von links)

Das Prinzip ist so erfolgreich, dass es von verschiedenen Entwicklern benutzt wird um eigene Distributionen daraus zu gestalten. (Siehe oben.) Vor allem das französische Projekt Linux Mint hat sich in kurzer Zeit zu einem Renner entwickelt und seine Quelle Ubuntu in der Beliebtheit überflügelt. Woran das liegt? Vielleicht in der übersichtlicheren Gestaltung des Desktops und der etwas besseren Möglichkeiten bei der Gestaltung seiner persönlichen Oberfläche.

Die Anwendungen sind das A und O

Aber eigentlich ist es egal, welche Farbe die Katze hat – Hauptsache, sie fängt Mäuse. Und die Mäuse sind für alle Systeme sehr ähnlich: wir wollen das Internet benutzen und möchten manchmal Texte schreiben oder in Tabellen rechnen. Musik sollte ordentlich anzuhören sein und Videos gucken auch. Manche möchten ihre Bildersammlung auf dem Rechner sortieren und vielleicht daraus eine Diashow gestalten. Andere suchen ein einfaches System, um die Kinder mit dem Rechner vertraut zu machen. Viele Wünsche und verschiedene Anforderungen. Aber für alles gibt es bewährte Lösungen für den Linux-PC.

Büroanwendungen

Alle kennen Microsoft Office, die ganze Welt arbeitet mit Word, Excel und Powerpoint. Die Namen dieser Anwendungen sind zu Begriffen in der Bürowelt geworden weil so ziemlich jeder damit schon einmal etwas zu tun gehabt hat. Selbst auf dem Mac wird die Office-Suite von Microsoft eingesetzt obwohl es dafür auch elegante Apple-Lösungen gibt. Auf Linux läuft Microsoft Office nicht ohne weiteres. Und das muss es auch nicht. Hierfür gibt es das genauso komfortable Paket LibreOffice.

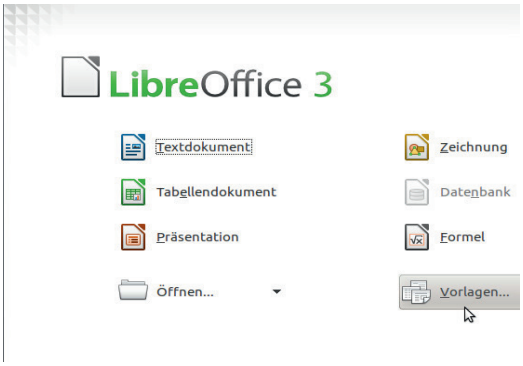


Bild 9 Libre Office

Word heißt hier Writer, was Excel kann macht hier die Anwendung Calc. Präsentationen gestaltet man mit Impress statt Power-Point und zum Malen gibt es Draw. Hat Microsoft so was zum Malen eigentlich auch?

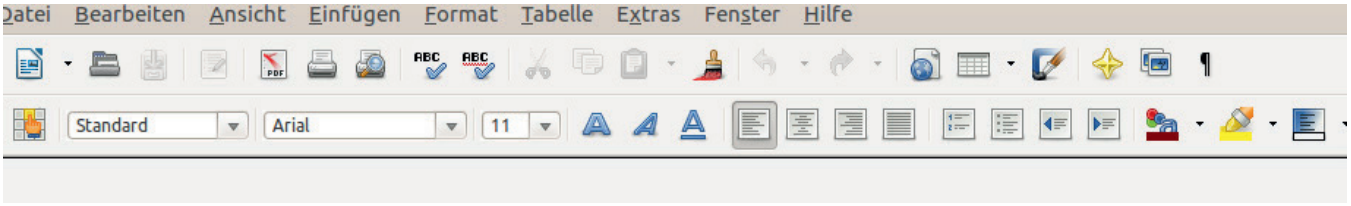


Bild 10: LibreOffice Writer

Das gute an diesen Anwendungen ist, dass sie alle Microsoft-Dokumente lesen können und dass man seinen Text auch als Word-Datei abspeichern kann, um sie den etwas zurückgebliebenen Kollegen hinter dem Windows-PC zuzuschicken. Versuchen Sie das mal umgekehrt.

Der Aufbau der Anwendungen ähnelt den Office-Programmen in der Version 2003. Umstellungsprobleme sollten es deshalb kaum geben. (Anders als die Umstellung von Office 2003 auf Office 2010.)

Musik und Videos

Das Ende der Schallplatten haben wir erlebt. Bei dem der CD sind wir mittendrin. Musik wird auf den PC kopiert oder gleich als Datei aus dem Internet gekauft und herunter geladen. Die Zeiten der eleganten CD-Racks im Wohnzimmer sind vorbei. Wer heute guckt, was er denn gern mal anhören möchte, der macht seine Musikverwaltung auf dem Rechner an und je nach dem wie gut diese sortiert ist, findet er Pink Floyd oder Santana oder besten Hits von 1974 und die Party beginnt. Für den Linux-Rechner gibt es dafür verschiedene Anwendungen. Sehr beliebt ist Banshee. Damit kann man nicht nur seine Musik hören, es gibt auch Schnittstellen zu Wikipedia und zu LastFM oder zu beliebigen Internetradios sowie Podcasts. Beliebige externe Hörgeräte z.B. Apple-iPod lassen sich anschließen. Sogar Videos kann man mit Banshee anschauen.

Wem die Gestaltung von Banshee nicht behagt, der verwendet den Rhythmbox oder probiert Amarok oder den für große Musiksammlungen gut geeigneten Gmusicbrowser.

Alles Geschmackssache wie ein Blick in die zahlreichen Foren zum Thema Unterhaltungsmedien belegt.

Apropos Forum: Ohne Forum geht heute fast nichts mehr. Auch die Ubuntuanwender finden Hilfe, Tipps und Anregungen im vorbildlichen Forum bei ubuntuusrs.de. Es gibt praktisch kein Problem, das nicht schon einmal aufgetreten ist und zu dem es nicht schon mal eine Frage samt Antworten gegeben hätte.

Zurück zu den Medienabspielern. Die Betriebssysteme haben natürlich schon einige Musik- und Video-Player dabei, die mit installiert

werden. Wer etwas anderes ausprobieren möchte, findet im reichhaltigen Software Center mit Sicherheit das richtige.

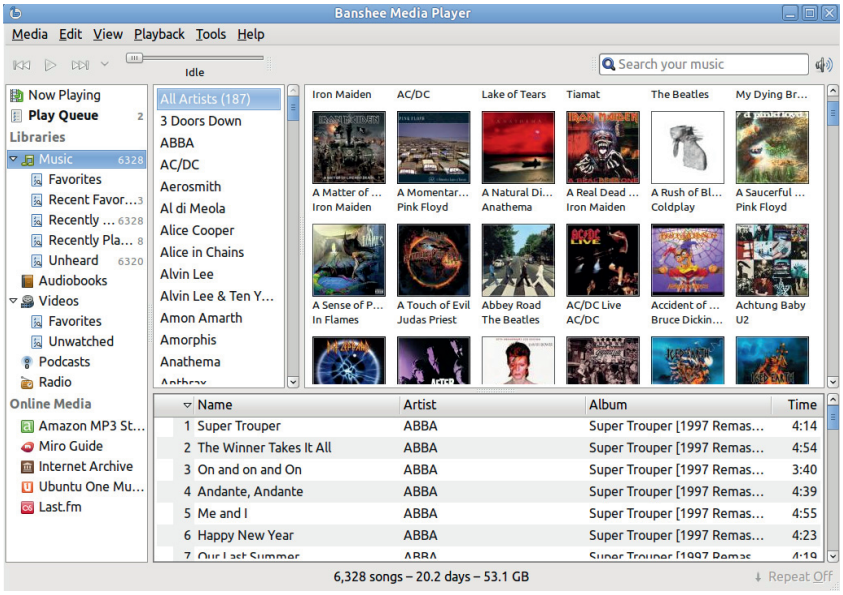


Bild 12 Software Center

Bilder sortieren und bearbeiten

Heute hat jedermann eine Digitalkamera und knipst drauflos. Filme entwickeln? Alles Vergangenheit, heute machen wir das alles selbst. Flugs werden die fotografischen Kunstwerke von der Kamera auf den Rechner geladen und da liegen sie dann und es werden immer mehr. Eine Bildverwaltung muss her. Bei Linux haben sich digiKam und Shotwell bewährt. Vor allem Digikam bietet vielfältige Möglichkeiten, seine Bilder zu verschlagworten und mit Geo-Informtionen zu versehen.

Auch eine kleine Bildbearbeitung ist integriert, die für die meisten Unfälle (grade ziehen, aufhellen, etwas wegschneiden) hinreicht. Die richtigen Halbprofis finden mit Gimp ein Programm, dass es mit Adobes Photoshop ohne weiteres aufnehmen kann. Die erweiterte Bildbearbeitung mit Ebenen und einer großen Zahl von Werkzeugen lassen kaum Wünsche offen.

Viele Kameras erzeugen Bilder im RAW-Format. Diese Bilder sind etwas größer als die üblichen jpg-Fotos und eignen sich für die verlustfreie Bearbeitung

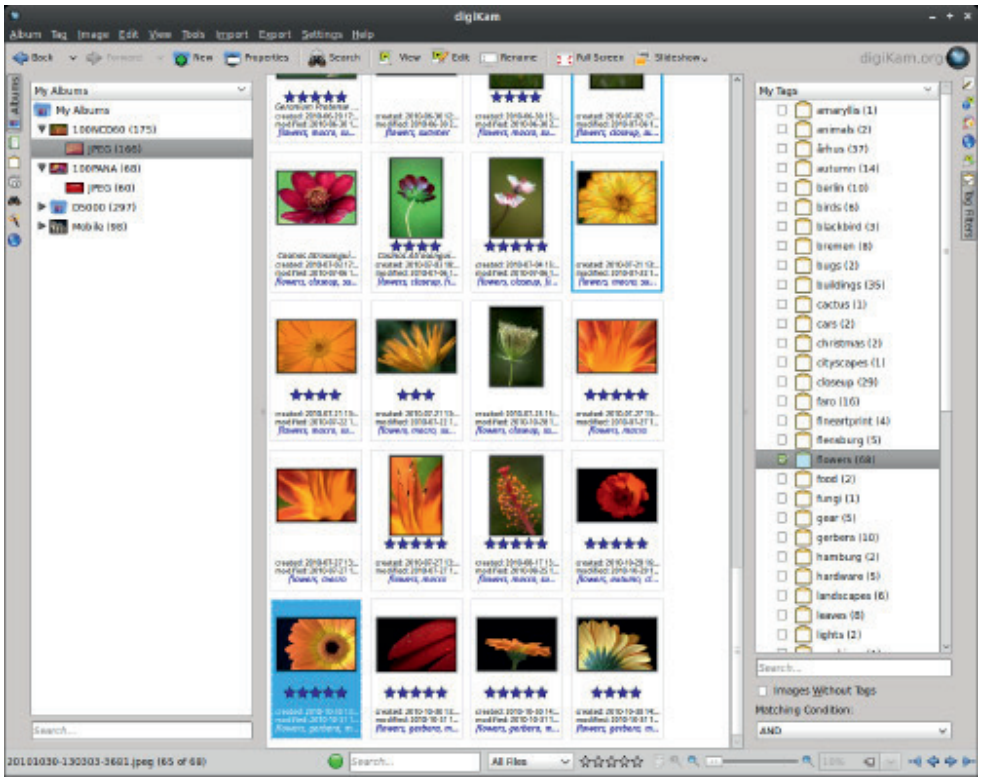


Bild 13 digiKam

Bild 14 Darktable RAW-Bildbearbeitung

mit einem entsprechenden Programm. Bei Linux gibt es dafür Rawtherapee und Darktable.

Beide Programme sind kostenlos und brauchen den Vergleich mit erheblich teureren Alternativen (Adobe Lightroom) nicht zu scheuen.

Fazit

Heute hat jeder mindestens einen PC und das eine oder andere Notebook, Netbook oder ein Tablet zu Hause. Die meisten Geräte werden mit Windows als Betriebssystem ausgeliefert. Und wem das reicht, den soll dieser kleine Artikel auch nicht in andere Gefilde weglocken. Es gibt schließlich keine „besseren“ Systeme. Womit man gut zurecht kommt, das ist das beste System. Erst wenn Schwierigkeiten auftauchen oder eine Neuanschaffung ansteht, sollte man über einen Wechsel nachdenken. Und vielleicht kommt dann auch Linux in die Auswahl. Es wäre nicht die schlechteste Entscheidung.

Kontakt:

Peter Thaben
Saubere Zeiten e.V.
Förderverein der Straßenreinigung und Müllabfuhr
Geschäftsstelle
Ringbahnstraße 96, D-12103 Berlin
eMail: info@saubere-zeiten.de

Jürgen Hanneder

Der TEXnische Fortschritt und seine Tücken

Nur zur Klarstellung: Wenn ich im folgenden über die Entwicklung von TEX et al. klage, dann ist dies die Klage eines ausschließlichen TEX-Benutzers, der immer noch nicht weiß oder wissen will, wie Formatvorlagen in Textverarbeitungsprogrammen funktionieren. Meine DANTE-Mitgliedsnummer ist sehr niedrig und meine Leidensfähigkeit, wenn es um Schrifsatz geht, ist ausgeprägt: Als ich als Student mit einem XT-Computer (Betriebssystem DOS) meine Abschlussarbeit formatierte, dauerte der TEX-Lauf meiner kritischen Edition eines Sanskrit-Textes mit edmac heute unvorstellbare drei Stunden (für edmac benötigt man aber drei Läufe!). Dennoch habe ich seither nichts anders als dieses genial Programm benutzt, den wenn man neun Stunden gewartet hatte, bekam man zumindest ein richtig gesetztes Buch, während die Konkurrenz sich mit Schreibmaschinenästhetik zufrieden geben musste. Manches hat sich zum Glück seither geändert. Derselbe TEX-Lauf – Ich habe es kürzlich mit der uralten Datei versucht –dauert nun unter einer Minute.

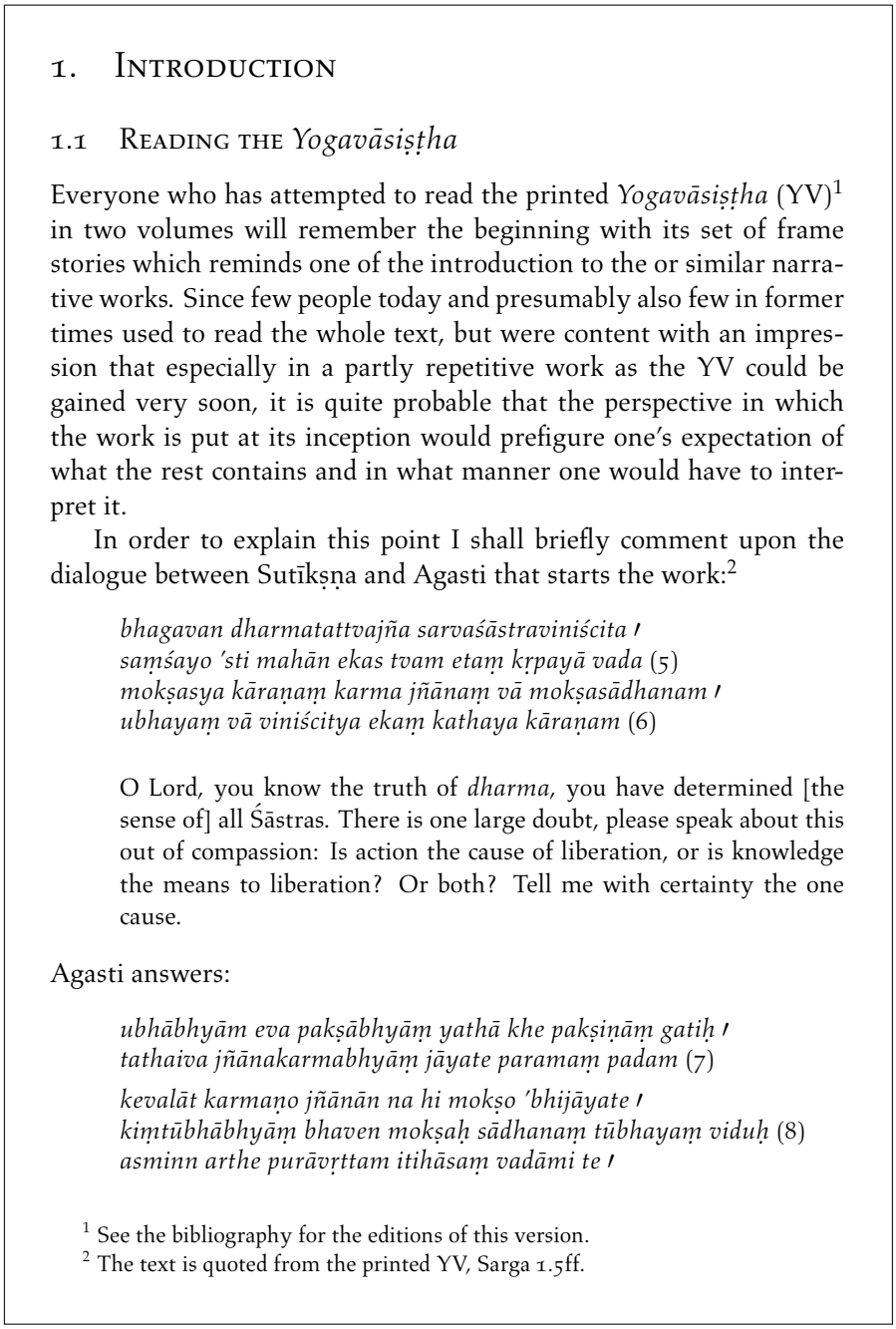
Auch anderes ist seither sehr viel komfortabler geworden. Anfangs mussten, wenn man indische Texte in Transkription setzen wollte, für diakritische Zeichen TEX-Kommandos eingeben werden. Den indischen Gott Krsna schrieb man als TEX-Anwender K\d{r}\d{s}\d{n}a. Man musste sich schließlich behelfen, indem man — für jedes System neu — einen Bildschirmfont, eine Tastaturbelegung und die TEX-Eingabekodierung schrieb, ein Problem, welches jede TEX-Implementation individuell löste. Doch die Vorteile lagen auf der Hand. Während die Redmond-hörigen Kollegen immer genervt nach einem Font mit allen Diakritika suchten, konnte man sich als TEX-Anwender entspannt zurücklehnen. Wollte man seine Spuren verwischen und nicht ein von weitem als solches erkennbares TEX-Dokument setzen, sondern ein normales Buch, dann musste man Computer Modern und die LATEX-Standardstile meiden.[1] Wenn man sich noch einen hochwertigen kommerziellen Font kaufte, der schon in TEX integriert war,[2] konnte man eigentlich sehr zufrieden sein.[3] Das größte Problem im Bereich der indischen Sprachen war das der Originalschriften. Hier — es geht im Folgenden aber nur um Sanskrit und Devanāgarī — hatte ein Indologe[4] schon längst eine gute Lösung entwickelt, die TEX

1 Siehe Beispiele auf S. 41 (Abhinavagupta's Philosophy of Revelation, Groningen 1998.)und S. 42 (Mokṣopāya. Kritische Edition. Wiesbaden 2011).

2 Siehe http://home.vr-web.de/was/fonts.

3 Siehe die Probeseiten aus indologischen Publikationen mit Aldus (S. 41) und Sabon (S. 42).

4 Siehe www.ctan.org/tex-archive/language/devanagari/velthuis



1. Introduction

1.1 Reading the *Yogavāsiṣṭha*

Everyone who has attempted to read the printed *Yogavāsiṣṭha* (YV)[1] in two volumes will remember the beginning with its set of frame stories which reminds one of the introduction to the or similar narrative works. Since few people today and presumably also few in former times used to read the whole text, but were content with an impression that especially in a partly repetitive work as the YV could be gained very soon, it is quite probable that the perspective in which the work is put at its inception would prefigure one's expectation of what the rest contains and in what manner one would have to interpret it.

In order to explain this point I shall briefly comment upon the dialogue between Sutīkṣṇa and Agasti that starts the work:[2]

bhagavan dharmatattvajña sarvaśāstraviniścita ।
saṃśayo 'sti mahān ekas tvam etaṃ kṛpayā vada (5)
mokṣasya kāraṇaṃ karma jñānaṃ vā mokṣasādhanam ।
ubhayaṃ vā viniścitya ekaṃ kathaya kāraṇam (6)

O Lord, you know the truth of *dharma*, you have determined [the sense of] all Śāstras. There is one large doubt, please speak about this out of compassion: Is action the cause of liberation, or is knowledge the means to liberation? Or both? Tell me with certainty the one cause.

Agasti answers:

ubhābhyām eva pakṣābhyāṃ yathā khe pakṣiṇāṃ gatiḥ ।
tathaiva jñānakarmabhyāṃ jāyate paramaṃ padam (7)

kevalāt karmaṇo jñānān na hi mokṣo 'bhijāyate ।
kiṃtūbhābhyāṃ bhaven mokṣaḥ sādhanaṃ tūbhayaṃ viduḥ (8)
asminn arthe purāvṛttam itihāsaṃ vadāmi te ।

[1] See the bibliography for the editions of this version.
[2] The text is quoted from the printed YV, Sarga 1.5ff.

Bild 1: Beispiel mit der Schrift Aldus (siehe Fn. 1)

sehr lange einen Wettbewerbsvorteil verschaffte. Zwar brauchte man einen Präprozessor, den man für jedes Betriebssystem kompilieren musste, und für die Eingabe wurde ein neues Transkriptionsschema geschaffen (Kṛṣṇa musste nun K.r.s.na geschrieben werden), doch das System wurde so beliebt, dass sich die vermutlich aus der Not erfundene Transkription sogar in der Indologie etablierte! Denn man konnte das, was die Konkurrenz heute noch nicht kann: Kritische Editionen in indischen Schriften drucken. Aber eigentlich nur in einem einzigem Schriftschnitt, denn man musste die Grenzen von TEX auf den verschiedensten Ebenen umgehen und alle Elemente auf die Struktur des Fonts zuschneiden. Da indische Schriften, wenn der Font nicht viele Hundert Zeichen haben soll, aus Einzelelementen zusammengesetzt werden, mussten alle diese Strichelemente möglichst effektiv in Fonts konventioneller Größe untergebracht und dann vom Programm zusammengesetzt werden. Man benötigte also für jeden Font einen nur darauf abgestimmten Präprozessor, da niemand versucht hatte, den Font mit TEX-Makros zusammenzusetzen, was sicher die für TEX-Fanatiker bessere Lösung gewesen wäre. Es gibt eine solche, wenn mich nicht alles täuscht, für die südindische Schrift Tamil.

Doch schien dies alles nicht mehr nötig, denn es winkte wieder der Fortschritt in Form neuer TEX-Varianten (Omega, Aleph etc.), die für Multilingualisten alles konnten, und für die man folgerichtig immer wieder Anpassungen produzierte.

Der Haken war, dass Forscher dies in ihrer Freizeit neben ihrer eigentlichen Forschung bewerkstelligen mussten. Doch diese Arbeit wurde immer wieder obsolet, da sich die TEX-Gemeinde nie auf *ein* besseres TEX konzentrieren wollte, und ich muss gestehen, dass ich mit steigendem Arbeitsaufwand die Geduld verlor, als Omega von Aleph, dann LuaTEX abgelöst wurde; sobald man mühevoll eine Anpassung an die aktuellen TEX-Programme erstellt hatte, gab es ein neues und die Entwicklung des alten stockte. Der gemeinsame Nenner schien die gegenüber dem Urprogramm immer weiter reduzierte Halbwertszeit zu sein, eine Entwicklung, die eigentlich den großen Wettbewerbsvorteil von TEX, seine spektakuläre Stabilität, zunichte machte. Natürlich: Wenn man englischen Text druckte, war alles kein Problem, aber wenn man Sanskrit drucken wollte, fing alles immer wieder von Neuem an. Ich muss für das Folgende die Perspektive daher noch weiter einengen: Es geht hier darum, TEX zu verwenden, um in einem begrenzten Zeitfenster ein Dokument zur Druckreife zu bringen. Vertröstungen auf die nächste Programmversion, welche die bisherigen Probleme elegant löst, werden plötzlich irrelevant. Was zählt, ist das, was zu einem fest definierten Zeitpunkt vollständig funktioniert — und zwar auch solche Probleme, die einem beim Herumprobieren nicht gleich auffallen und die wahrscheinlich dem Entwickler, der seine Makros nie zur Buchproduktion verwendet

PRATHAMAS SARGAḤ

vasiṣṭhaḥ

vāgbhābhir brahmavid brahma bhāti svapna ivātmani
yad idaṃ tat svaśabdārthair yo yad vetti sa vettu tat 1.1

nyāyenānena sarvasmin sarge brahmāmbare sati
kim idaṃ kasya vakṣīti codyacañcur nirākṛtaḥ 1.2

ahaṃ tāvad yathājñānaṃ yathāvastu yathākramam
yathāsvabhāvaṃ vacmīdaṃ tat sarvaṃ śrūyatāṃ budhāḥ 1.3

svapnavat paśyati jagac cinnabho dehavinmayam
svapnasaṃsāradṛṣṭāntā ihaivāntas samanvitāḥ 1.4

mumukṣuvyavahāroktimayāt prakaraṇād anu
athotpattiprakaraṇaṃ mayedaṃ parikathyate 1.5

bandho 'yaṃ dṛśyasadbhāvo dṛśyābhāve na bandhanam
na sambhavati dṛśyaṃ tu yathedaṃ tac chṛṇu kramāt 1.6

utpadyate yo jagati sa eva kila vardhate
sa eva mokṣam āpnoti svargaṃ vā narakaṃ ca vā 1.7

atas te svavabodhārthaṃ tat tāvat kathayāmy aham
utpattiṃ saṃsṛtāv eti pūrvam eva hi yo yathā 1.8

idaṃ prakaraṇārthaṃ tvaṃ saṅkṣepāc chṛṇu rāghava
tataḥ prakathayiṣyāmi vistaraṃ te yathepsitam 1.9

yad idaṃ dṛśyate sarvaṃ jagat sthāvarajaṅgamam
tat suṣupta iva svapnaḥ kalpānte pravinaśyati 1.10

tatas stimitagambhīraṃ na tejo na tamas tatam
anākhyam anabhivyaktaṃ yat kiñcid avaśiṣyate 1.11

1a $Ś_{14}$ 58^{r} 1ab brahma bhāti svapna ivātmani cett.] *bra*[…]*ivātmani* $Ś_{14}$ 1d vettu cett.] *vetti* $Ś_7$ 2ab sarvasmin sarge cett.] […]*rge*[…] $Ś_{14}$ 2b brahmāmbare sati cett.] *brahmākāmbare*[…]*ati* $Ś_{14}^{pc\,s.m.}$ 2c vakṣīti $Ś_1^{pc}Ś_3Ś_7Ś_9N_{12}N_{20}$] *vakṣeti* $Ś_1^{ac}$, l.n. $Ś_{14}$ 2d codya cett.] […] $Ś_{14}$ 2d cañcur cett.] *cuñcur* $Ś_9$, […]*uñcur* $Ś_{14}$ 3c yathāsvabhāvaṃ cett.] *yathā*[…] $Ś_{14}$ 3c vacmīdaṃ $Ś_1Ś_3Ś_7Ś_9N_{12}$] *vaśmīdaṃ* N_{20}, […]*īdaṃ* $Ś_{14}$ 4a paśyati $Ś_1Ś_7Ś_9Ś_{14}^{ac}N_{12}N_{20}$] *paśyata* $Ś_3$, *yūśaṃta* $Ś_{14}^{pc\,s.m.}$ 4c svapnasaṃsāradṛṣṭāntā cett.] *sva*[…]*ṣṭāntā* $Ś_{14}$ 4d ihaivāntaḥ $Ś_1N_{12}N_{20}Ś_9Ś_{14}$] *ihaivātas* $Ś_3Ś_7$ 4d $Ś_9$ 98^{r} 5b anu $Ś_7Ś_{14}$] *param* $Ś_1Ś_3Ś_9N_{12}N_{20}$ 5c athot cett.] […]*t* $Ś_{14}$ 6d tac chṛṇu $Ś_7Ś_9Ś_{14}N_{12}$] *śṛṇu tat* $Ś_1Ś_3N_{20}$ 7d N_{20} 2^{r} 8a te svava $Ś_1Ś_3Ś_7Ś_9Ś_{14}^{ac}N_{20}$] *teṣv ava* N_{12}, *te svāva* $Ś_{14}^{pc\,s.m.}$ 8c pūrvam cett.] *p*[…]*rvam* $Ś_{14}$ 9a idaṃ $Ś_7Ś_9Ś_{14}N_{12}$] *imaṃ* $Ś_1Ś_3N_{20}$ 10d pravinaśyati cett.] *pravinanaśyati* $Ś_1^{ac}$

Bild 2: Beispiel mit der Schrift Sabon (siehe Fn. 1)

hatte, ebenfalls nicht bewusst waren. Steht man vor dieser Aufgabe, so merkt man, dass die TEX-Gemeinde mehr entwickler- als anwenderorientiert ist. Es gibt viele Lösungen für Probleme, jedoch vergleichsweise wenige gelungene Anwendungsbeispiele, zumindest für die Geisteswissenschaften. Es gibt ja nicht einmal eine Dokumentklasse, die standardmäßig ein Seitenformat nahe am Goldenen Schnitt produziert, was natürlich auch daran liegt, dass die meisten TEX-Anwender vermutlich aus den Naturwissenschaften kommen und dort die DIN-Formate beliebt sind. Es ist kein Problem DIN B5 zu drucken, aber ein schönes Format für die Wissenschaftliche Buchgesellschaft muss man sich selbst schreiben. Natürlich ist so etwas nicht schwierig, doch der erzieherische Effekt wird vertan. Wir wollen ja auch vermeiden, dass sich falsche Kapitälchen aufgrund der Sehgewohnheiten als Standard einprägen. Dass auch viele Wissenschaftsverlage nur noch DIN B5 zu kennen scheinen, erschwert die Sache erheblich, aber ein Gang in die Belletristikabteilung einer größeren Buchhandlung kann hier heilsam sein. Dort gibt es keine DIN-Formate, doch das nur am Rande.

Die entscheidende Entwicklung war die Erweiterung der Zeichensätze. Die vermutlich englischsprachigen Fontdesigner wussten zwar, dass man in Europa kleine Punkte und Striche über bestimmte Buchstaben setzt, die heute gelegentlich der OCR-Software zum Opfer fallen, haben den Sinn dieser alten europäischen Sitte aber lange nicht eingesehen. Jetzt geht man davon aus, dass moderne Unicode-Zeichensätze alle Diakritika beherrschen, doch ist dies eben nur fast richtig, denn der Großteil der Unicode-Fonts kann zwar einen Teil der für die Umschrift von Sanskrit verwandten Zeichen (unproblematisch ist ā, ū und ī), aber nicht alle (viele scheitern an ṭ, ṇ oder ṛ). Stellen Sie sich also die Situation vor, dass die meisten Zeichensätze zwar ä und ü können, aber kein ö. Die theoretisch sehr hohe Zahl an Möglichkeiten schwindet. Auch Hinweise auf Fonts, die entweder nicht weiterentwickelt und nicht mehr bereitgestellt werden (Code2000) oder solche, deren Copyright eine kommerzielle Verwendung ausschließen (TitusCyberbit), erweisen sich für den Praktiker als Sackgassen, so interessant sie zum Experimentieren im einzelnen auch sein mögen.

Haben Sie sich bei dem Gedanken ertappt: Aber es gibt doch jetzt XƎTEX (XeTeX), das löst doch solche Probleme schon längst! Die natürliche Reaktion auf Probleme mit dem alten Programm ist bekanntlich, eine neue Version zu installieren. Wir sind und bleiben wohl technikbegeisterte Kinder des Fortschritts. Und natürlich habe auch ich immer die letzte Linux-Version auf allen meinen Rechnern und teste seit längerem XƎTEX, drucke Devanagarl damit, habe mich aber erst jetzt, sozusagen unter Sachzwang, dazu durchringen müssen, ein Buch damit zu setzen. Der Grund liegt in der zwiespältigen Natur der Neuerungen dieses Programmes. Einerseits ist es phantastisch, dass nun alle Systemschriften mit TEX kooperieren (oder umgekehrt). Aber die Zahl der Schriften, die sich für den wissenschaftlichen Buchdruck in der Indologie wirklich eignen, vergrößert sich weitaus weniger spektakulär. Die Ausgangslage war, dass in TEX seit jeher die Eingabe \d{d} das richtige Zeichen (ḍ) ergibt und man

den TEX-Code unter einer besser lesbaren Oberfläche verbergen kann. Man hat einen Bildschirmfont mit den richtigen Zeichen, die Eingabekodierung setzt dies in ḍ um, ein Tastaturmakro erleichtert die Eingabe. Und der zum Druck verwandte Font steht unter der Kontrolle von TEX so dass die große Zahl der mit TEX kooperierenden Zeichensätzen keine Probleme macht. Dieses Verfahren wurde in DOS-Zeiten noch

Devanāgarī unter XHTEX

Die folgende Übersicht zeigt die mir bekannten Devanāgarī-Zeichensätze, die bisher in der Praxis mit XHTEX zusammenarbeiten.

फेरेङ्गितीर्थसङ्काशे काशति पुरी सुन्दरी
मारदुर्गाभिधाना सा लानानूपे व्यवस्थिता — *Siddhanta*

फेरेङ्गितीर्थसङ्काशे काशति पुरी सुन्दरी
मारदुर्गाभिधाना सा लानानूपे व्यवस्थिता — *Samyak*

फेरेङ्गितीर्थसङ्काशे काशति पुरी सुन्दरी
मारदुर्गाभिधाना सा लानानूपे व्यवस्थिता — *Nakula*

फेरेङ्गितीर्थसङ्काशे काशति पुरी सुन्दरी
मारदुर्गाभिधाना सा लानानूपे व्यवस्थिता — *Sahadeva*

फेरेङ्गितीर्थसङ्काशे काशति पुरी सुन्दरी
मारदुर्गाभिधाना सा लानानूपे व्यवस्थिता — *Sanskrit2003*

फेरेङ्गितीर्थसङ्काशे काशति पुरी सुन्दरी
मारदुर्गाभिधाना सा लानानूपे व्यवस्थिता — *Jaipur*

फेरेङ्गितीर्थसङ्काशे काशति पुरी सुन्दरी
मारदुर्गाभिधाना सा लानानूपे व्यवस्थिता — *JanaSanskrit*

फेरेङ्गितीर्थसङ्काशे काशति पुरी सुन्दरी
मारदुर्गाभिधाना सा लानानूपे व्यवस्थिता — *shiDeva*

Bild 3: Devanāgarī-Fonts für Sanskrit

mit emTEX realisiert, dann mit Omega etc. Eingabeseitig hat sich nicht viel geändert, man benötigt immer noch einen geeigneten Bildschirmzeichensatz — viele, die behaupten, unicode-fähig zu sein, können ḍ nicht darstellen — und eine Eingabemethode. Doch mit XƎTEX — und dies ist der entscheidende Unterschied — hat man die Kontrolle über die Diakritika den Fontdesignern übertragen, die selten Indologen sind. Was bedeutet dies für den Praktiker? Ganz einfach: Die theoretische Vielfalt an Fonts, die XƎTEX so reizvoll macht, schmilzt im Nu auf die bisherige Größe zusammen. Die meisten Fonts, die mit XƎTEX zusammenarbeiten, stellen die Umschrift des Sanskrit nicht korrekt dar. Bei manchen fehlen Zeichen (ḍ ist wieder beliebt), bei anderen stimmt das Kerning nicht usw. Sicher kleine, vielleicht auch leicht behebbare Fehler (wenn man Fontdesigner ist), aber solche, die den Font für eine Buchproduktion ausschließen. Da die hochwertigen kommerziellen Fonts, die man sich für die alten TEX-Versionen gekauft hatte (Type 1, kein OpenType), nicht mehr funktionieren, fängt man wieder neu an, kauft Fonts und hofft, dass sie gut programmiert sind. Ich werde für die genannte Publikation dennoch XƎTEX verwenden, denn erstmals mit TEX eine Auswahl aus mehreren Devanāgarī-Schriften zu haben, ist ein durchaus erhebendes Gefühl. Aber auch hier gilt: Nur ein Bruchteil funktioniert wirklich, und ich bin — um es positiv auszudrücken — immer wieder fasziniert, wie wenig Anspruch und Realität zusammenpassen. Manche haben typographische Mängel, andere sind vermutlich unzureichend programmiert und verschlucken ganze Ligaturen. Kein einziger der anlässlich der World Sanskrit Conference in New Delhi im Januar 2012 von indischen Entwicklern verteilten Zeichensätze tut, was er soll (zumindest mit XƎTEX)! Kommerzielle Devanāgarī-Fonts für Sanskrit sind bisher ebenfalls Fehlanzeige, wobei URW kurz vor der Fertigstellung eines solchen steht. Bild 3 zeigt die acht Zeichensätze, die nach einer langwierigen Sichtung einer großen Zahl übrig geblieben sind.

Versucht man nun in einer Publikation, europäische und indische Schriften in ein und demselben Dokument beziehungsweise in derselben Zeile zu setzen, dann müssen »Ost« und »West« auch noch bei Schriftdicke und Stil zusammenpassen. Dadurch fällt einer der beliebtesten Sanskrit-Fonts (Sanskrit2003) wieder heraus, da er eigentlich die fette Variante einer nicht existierenden Normalschrift ist. Es bleiben also weitaus weniger Möglichkeiten, als man denkt, aber am Ende — und daher will ich mit einer guten Nachricht enden — lässt sich mit XƎTEX das Buch dann doch so produzieren, wie ich es mir vorstelle, denn der Devanāgarī-Font Siddhanta passt glücklicherweise ganz gut zum TEX-Font Libertine. Aber ich experimentiere noch.

Kontakt:

Universitätsprofessor Dr. Jürgen Hanneder
Institut für Klassische Sprachen und Literaturen, Fachgebiet Indologie und Tibetologie
Philipps-Universität Marburg
Deutschhausstraße 12, 35032 Marburg
eMail: juergen.hanneder@staff.uni-marburg.de
Tel. 06421.28 24930

Anschluss an die Spitzengruppe der TOP500-Supercomputer

Mit SuperMIG und SuperMUC nimmt das Leibniz-Rechenzentrum neue Höchstleistungsrechner in Betrieb

München/Garching. Ende Juni 2011 wurde am Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) der Höchstleistungsrechner „SuperMIG“ installiert, der zu den 500 leistungsfähigsten Rechnern der Welt zählt. Bei „SuperMIG“ handelt es sich um das Migrationssystem für den Wechsel vom bisherigen Höchstleistungsrechner „HLRB II“ auf den nächsten, von IBM entwickelten Höchstleistungsrechner „SuperMUC“(Inbetriebnahme Juli 2012). In der „Top500-Liste“ rangiert SuperMIG auf Platz 166. SuperMIG ist ein IBM Blade Center System mit 8200 Rechenkernen und 52 Terabyte Hauptspeicher. Das System wird in den neuen „SuperMUC“ integriert, um dauerhaft Anwendern, die hohe Hauptspeicherkapazitäten benötigen, zur Verfügung zu stehen.

Eines der Hauptprobleme in der Entwicklung der Rechnertechnik ist der Energieverbrauch. SuperMIG erreicht eine gegenüber dem HLRB II um etwa 10 Prozent höhere Leistung bei einem Stromverbrauch von nur noch etwa einem Fünftel. Torsten Kurz, IBM Deutschland, erklärte hierzu: „Wir legen bei der Entwicklung unserer Höchstleistungsrechner größtes Augenmerk auf die Energieeffizienz. SuperMUC wird gegenüber dem jetzigen Migrationssystem eine weitere, drastische Steigerung der Rechenleistung bei noch deutlich niedrigerem Energieverbrauch bringen. Wir werden dies durch eine innovative Kühlung der Hardware, die beispielsweise Prozessoren und Hauptspeicher mit warmen Wasser kühlt, sowie eine intelligente Steuerung des Betriebs, die an Energieeffizienzaspekten ausgerichtet ist, erreichen.“

Bei dieser eigens von IBM entwickelten Warmwasserkühlung, die im SuperMUC erstmals in großtechnischem Maßstab eingesetzt wird, werden die Prozessoren und der Hauptspeicher direkt mit bis zu 55 Grad Celsius warmem Wasser gekühlt, es sind keine zusätzlichen Kühlwerke nötig.

Mit mehr als 3 Petaflops Rechenleistung (also drei Billiarden Rechenoperationen pro Sekunde oder eine 3 mit 15 Nullen) wird SuperMUC der viertschnellste Rechner der Welt und der schnellste Rechner Europas sein. Er beruht auf dem System x iDataPlex von IBM, verfügt über mehr als 155.000 Rechenkerne und mehr als 330 Terabyte Hauptspeicher, die über ein nicht-blockierendes InfiniBand-Netzwerk mit Fat Tree-Topologie kommuniziert werden können. Darüber hinaus können bis zu 10 Petabyte Daten in einem parallelen GPFS-Dateisystem von IBM

SuperMUC: Mit mehr als drei Petaflops Rechenleistung einer der leistungsstärksten Rechner weltweit (Foto: LRZ)

zwischengespeichert werden. Für die dauerhafte Speicherung der Benutzerdaten wie Programmquellen und Eingabedatensätzen steht eine Speicher-Lösung von NetApp mit einer Kapazität von 4 Petabyte zur Verfügung, die sich durch hohe Zuverlässigkeit auszeichnet. Zusätzlich stehen für die langfristige Archivierung von Daten des SuperMUC 16,5 Petabyte Archivkapazität auf Bandsystemen zur Verfügung.

SuperMIG und SuperMUC werden vom Freistaat Bayern und dem Bundesministerium für Bildung und Forschung kofinanziert und stehen nicht nur Wissenschaftlern in Deutschland, sondern auch aus 21 europäischen Ländern in der Partnership for Advanced Computing in Europe (PRACE) zur Verfügung.

Das Leibniz-Rechenzentrum (LRZ) der Bayerischen Akademie der Wissenschaften auf dem Forschungscampus in Garching ist der Dienstleister auf dem Gebiet der Informationsverarbeitung für die Münchner Hochschulen. Es stellt mit dem Münchner Wissenschaftsnetz (MWN) eine leistungsfähige Kommunikationsinfrastruktur für die Wissenschaften bereit und betreibt umfangreiche Datensicherungssysteme (Archivierung und Backup). Darüber hinaus ist das LRZ unter der Leitung von Professor Dr. Arndt Bode nationales Supercomputing Centre und Teil des Gauss Centre for Supercomputing, das von den drei nationalen Höchstleistungsrechenzentren (Garching, Jülich, Stuttgart) gebildet wird.

Kontakt: Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Boltzmannstraße 1, 85748 Garching bei München, Telefon: (089) 35831 8000, Fax: (089) 35831 9700, eMail: presse@lrz.de, Internet: http://www.lrz.de, http://www.lrz.de/services/compute/supermuc/systemdescription/

APL-Journal
30. Jg. 2011, ISSN 1438-4531
Herausgeber: Dr. Reiner Nussbaum, APL-Germany e.V., Mannheim, http://www.apl-germany.de
Redaktion: Dipl.-Volksw. Martin Barghoorn (verantw.), TU Berlin, Franklinstr. 28, D-10587 Berlin, Tel. (030) 314 24392, Fax (030) 314 25901
Verlag: RHOMBOS-VERLAG, Berlin, Kurfürstenstr. 17, D-10785 Berlin, Tel. (030) 261 9461, Fax (030) 261 6300, eMail: verlag@rhombos.de, Internet: www.rhombos.de
Erscheinungsweise: halbjährlich
Erscheinungsort: Berlin
Satz: Rhombos-Verlag
Druck: dbusiness.de GmbH, Berlin
Copyright: APL Germany e.V. (für alle Beiträge, die als Erstveröffentlichung erscheinen)
Fotonachweis Titelseite: Martin Barghoorn

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutzgesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen. Eine Haftung für die Richtigkeit der veröffentlichten Informationen kann trotz sorgfältiger Prüfung von Herausgeber und Verlag nicht übernommen werden. Mit Namen gekennzeichnete Artikel geben nicht unbedingt die Meinung des Herausgebers oder der Redaktion wieder. Für unverlangte Einsendungen wird keine Haftung übernommen. Nachdruck ist nur mit Zustimmung des Herausgebers sowie mit Quellenangabe und Einsendung eines Beleges gestattet. Überarbeitungen eingesandter Manuskripte liegen im Ermessen der Redaktion.

Allgemeine Informationen

(Stand 2011)

APL-Germany e.V. ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache APL, sowie die Verbreitung des Verständnisses der Mensch-Maschine Kommunikation zu fördern. Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen, sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet APL-Germany den adäquaten organisatorischen Rahmen.

Auf Antrag, über den der Vorstand entscheidet, kann jede natürliche oder juristische Person Mitglied werden. Organe des Vereins sind die mindestens einmal jährlich stattfindende Mitgliederversammlung sowie der jeweils auf zwei Jahre gewählte Vorstand.

1. Vorstandsvorsitzender
Dr. Reiner Nussbaum
Dr. Nussbaum gift mbH, Buchenerstr. 78, 69259 Mannheim,
Tel. (0621) 7152190.

2. Vorstandsvorsitzender:
Martin Barghoorn
Sekr. FR 6-7, Technische Universität Berlin, Franklinstr. 28,
10587 Berlin,
Tel. (030) 314 24392

Schatzmeister
Uwe Janza
Janza Software Consulting,
Mittlere Kanalstraße 4a
90429 Nürnberg
Tel. 0911-327075,
Fax -327076

Ordentliche Mitglieder:

Natürliche Personen	32,- EUR*
Studenten / Schüler	11,- EUR*

Außerordentliche Mitglieder:

Jurist./natürl. Pers.	500,- EUR*

* Jahresbeitrag

Bankverbindung
BVB Volksbank eG Bad Vilbel
BLZ 518 613 25
Konto-Nr. 523 2694

Hinweis:
Wir bitten alle Mitglieder, uns Adressänderungen und neue Bankverbindungen immer sofort mitzuteilen. Geben Sie bei Überweisungen den namen und/oder die Mitgliedsnummer an.

Einzugsermächtigung

Mitglieds-Nr.: _____

Ich erkläre mich hiermit widerruflich damit einverstanden, daß APL Germany e.V. den jeweils gültigen Jahres-Mitgliedsbeitrag von meinem unten angegebenen Konto abbucht. Mit der Datenübermittlung an das oben genannte Kreditinstitut bin ich einverstanden. Einen eventuell bestehenden Dauerauftrag habe ich bei meiner Bank gelöscht.

Bankbezeichnung: _____

BLZ: _____ Konto-Nr.: _____

Datum: _____ Unterschrift: _____

NEWS:
Neues@APL-Germany.com

WEB:
www.apl-germany.de