Wie Jumbos ausgelastet werden ...

Eine Analyse von Gullivers neuester Reise ins Land der Riesen-Computer[1]

Von J. Swift jun.

[1] Erstveröffentlichung in: Online ZfD Zeitschrift für Datenverarbeitung 4/1974, 228 - 230

Das Problem ist schon lange bekannt. Jumbo-Computer, wie man die größten Rechner der EDV mit Hochachtung benennt, stellen sowohl die Mitarbeiter in den Fachabteilungen als auch in Rechenzentren vor Auslastungsschwierigkeiten. Zumindest gilt das in der Phase der Einarbeitung, oft aber auch während der gesamten Installationsdauer. So ist denn mit Sicherheit eine Programmiertechnik wertvoll, wie sie diese Analyse ebenso nüchtern wie mit wissenschaftlichem Ernst beschreibt: Den sichersten Weg zum Rund-um-die-Uhr-Betrieb und damit zur wirtschaftlichen Auslastung weist die PP-Technik.

Die Programm-Pessimierungs-Technik (PP-Technik) ist im eigentlichen Sinne keine wirklich neue Entwicklung. Einen ersten Hinweis fand der Verfasser schon in der amerikanischen Literatur. Während sich jedoch jener Artikel (1) mehr mit dem Übersetzer-theoretischen Aspekt der Methode befaßt hat, soll hier auf die praktischen Anwendungsmöglichkeiten hingewiesen werden, die diese Methode bietet. Damit soll das praktische Verständnis erleichtert und die Methode einem größeren Kreis von EDV-Benutzern nahegebracht werden.

Noch vor wenigen Jahren war es überflüssig, sich über die Art und Weise des Programmierens Gedanken zu machen. In Zukunft wird man jedoch nicht auf das Nachdenken über das Programmieren verzichten können. Zwar existieren schon Gedanken, die in diese Richtung gehen (etwa normierte Programmierung und Programmgeneratoren); diese sind jedoch nicht so wirkungsvoll, wie die PP-Technik.

Jeder, der sich mit Programmierung befaßt, wird bereits einige Regeln der PP-Technik angewandt haben, wenn auch bewußt nur zu Beginn seiner Programmiertätigkeit. Die Tatsache, daß diese Regeln später nur unbewußt angewandt werden, zeigt deutlich, wie hier ursprüngliche Fähigkeiten verschüttet werden.

Früher waren die DV-Abteilungen und Rechenzentren mit EDV-Systemen ausgestattet, die nur einen einzigen Prozessor besaßen und meist im Mono-Programm-Betrieb genutzt wurden. Das heute erweiterte Instrumentarium kann nur noch durch bewußte und intensive Anwendung der Verfahren und Regeln der PP-Technik beherrscht werden.

Die PP-Technik in Theorie und Praxis

Grundsätzlich sind die vorausgesetzten Kenntnisse zur Anwendung der PP-Technik sowohl im Hinblick auf Hardware als auch Software so gering, daß sie hier nicht genannt zu werden brauchten. Allerdings kann dem Anwender der PP-Technik und auch dem Anfänger im Programmieren nur dringend geraten werden, die Anforderungen an die Hardware- und Software-Konfiguration des gesamten Systems so hoch wie möglich anzusetzen. Diese Forderung läßt sich einfach begründen.

Das Hardware-System sollte mit allen auf dem Markt erhältlichen Typen von Peripheriegeräten ausgestattet sein. Denn nur dann kann der Programmierer auch alle ihm zur Verfügung stehenden Befehle voll ausnutzen. Sind nicht alle Peripheriegeräte vorhanden, so bleiben Befehle ungenutzt. Auch werden Befehle nicht voll ausgenutzt, wenn ein- und derselbe Befehl auf mehrere Geräte Anwendung finden könnte. Beides kann nicht den Intentionen der PP-Technik entsprechen.

Der PPU (PP-Technik User) sollte darauf achten, daß alle verfügbaren Geräte on-line an das EDV-System angeschlossen sind. Dieser Gedanke ist noch lange nicht bei allen EDV-Installationen verwirklicht worden. Als Beispiel sei nur auf die off-line Installation von Druckern und Plottern hingewiesen: etwa 50% aller Plotter sind noch off-line angeschlossen.! Begründet wird die off-line Installationen von Peripheriegeräten mit der geringeren Systembelastung. Ein Ziel des Programmierens mit Hilfe der PP-Technik ist jedoch gerade die größere Auslastung und damit Belastung des Systems.

Entsprechendes gilt für den Software-Anteil des EDV-Systems. Auch hier sollte es die Aufgabe jedes PPU sein, auf nicht vorhandene Software hinzuweisen und deren Implementation zu erzwingen. Dieser Hinweis gilt nicht nur für vom Hersteller erhältliche kostenlose oder kostenpflichtige Software, sondern er bezieht sich auch auf von Software-Häusern erhältliche Programmsysteme.

Auf die Problematik im Einsatz dieser Software wird weiter unten noch eingegangen.

In jedem Fall sollten jedoch sämtliche vom Hersteller erhältlichen Komponenten des Betriebssystems implementiert sein und ständig hauptspeicherresident gehalten werden. Das gilt insbesondere für zur Zeit auf der vorhandenen Installation noch nicht nutzbare Komponenten, wie etwa Prozeduren zur Datenfernverarbeitung. Die vehemente Entwicklung der Datenverarbeitung wird auch an der Installation des Lesers nicht spurlos vorübergehen. Auch wenn gerade eine Entscheidung gegen den Einsatz der Datenfernverarbeitung gefallen sein sollte, so empfiehlt es sich trotzdem, die entsprechenden Betriebssystemkomponenten zu implementieren und mitlaufen zu lassen. Und sei es auch nur, um gegen alle Eventualitäten abgesichert zu sein.

Ähnliches gilt im Bereich der Anwendungssoftware. Alle Anwendungsprogramme sollten ständig auf den Bibliotheksplatten katalogisiert sein. Diese Forderung gilt insbesondere für die seltener benutzten Programme, Programmsysteme und Compiler. Gerade die selten oder nie benutzte Software muß ständig im direkten Zugriff erreichbar sein. Wie anders könnte sich der Benutzer sonst kurzfristig über ihr Vorhandensein informieren.

Der Benutzer-spezifische Aspekt der Programmiermethode

Hier soll vor allem auf problemorientierte Programmiersprachen eingegangen werden. Dem Verfasser erscheint es sinnvoll, anhand einiger konkreter und im täglichen Programmierbetrieb auch anzuwendender Beispiele die Verfahrensweise der PP-Technik zu erläutern.

Allerdings soll nicht auf grundlegende Prinzipien eingegangen werden, die sich unter den Datenverarbeitern weitgehend durchgesetzt haben, wie etwa die Entwicklung von Software-Systemen (Abb.) oder das Austesten von Programmen und Programmsystemen in ihrer Gesamtheit (statt bausteinweise oder in modularer Weise).

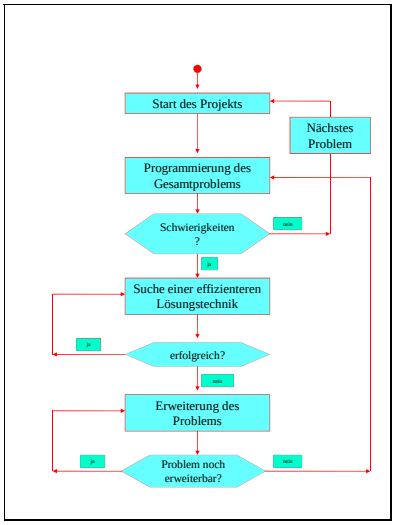

Abb.: Entwicklung von Softwaresystemen mit der PP-Technik

Gerade der Aufbau und das Austesten von Programmen in ihrer Gesamtheit und damit in nicht-modularer Weise zeigen auf jeder Stufe der Programmierung, Entwicklung und des Tests die Komplexität des Systems auf und ermöglichen damit eine Übersicht über die vorhandenen logischen und seman-

tischen Fehler. Beim modularen Aufbau und beim modularen Testen wird die Vielfalt der logischen und semantischen Fehler weitgehend verdeckt, da hier ja in kleinlicher Weise ein Fehler nach dem anderen aufgespürt werden muß. Das Bearbeiten von Programmiersystemen in ihrer Gesamtheit zeigt also deutlich einen großzügigen Rahmen auf.

Eine Alternative zur nicht-modularen Programmiermethode ist nicht denkbar.

Da sich der Übergang von der modularen zur nicht-modularen Programmierung nicht kurzfristig durchsetzen lassen wird, soll hier auf eine Vorgehensweise hingewiesen werden, deren Auswirkungen denen der PP-Technik durchaus nicht nachstehen.

In jüngster Zeit scheint sich folgende Methode durchzusetzen, die sich vor allem bei großen Programmsystemen (Compilern usw.) lohnt:

Das Programm wird modular aufgebaut. Eine Grundphase (root) steuert den Programmablauf und lädt benötigte Phasen nach. Tritt beim Ablauf des Programms ein Fehler auf, so wird ein dump-modul nachgeladen und auf die Grundphasen gelegt.

Die Grundphase wird kurz gehalten und sehr genau ausgetestet. Die eigentlichen Verarbeitungsphasen werden unausgetestet freigegeben. Die beim Fehlerabbruch generierten dumps übergibt der Benutzer dem Programmierteam zur Fehlersuche.

Das Testen des Programmsystems wird in einen Rückkopplungsprozeß eingebaut. Häufig benutzte Phasen und Modulen werden in kürzester Zeit ausgetestet; selten oder nie benutzte bleiben unausgetestet. Der zeitliche Aufwand des Programmierteams für Tests wird damit auf ein Minimum beschränkt.

Stellt man ein und dasselbe Programm zweimal im Kernspeicher bereit, so kann man im Fehlerfall immer noch auf das Duplikat zurückgreifen und weiterarbeiten. Dieses Vorgehen verhindert beim Benutzer den Eindruck, das Programm sei mit Fehlern behaftet.

Werden Konstante erneut zu jedem Lesen eines Satzes oder Blockes benötigt, so empfiehlt es sich, sie vor jedem Lesen auch erneut zu setzen. Da derartige Eingabeoperationen heutzutage leider noch oft in Schleifen abgearbeitet werden statt in sequentieller Folge, ist es sinnvoll, Anfangsanweisungen in die Schleife hineinzunehmen. Diese Maßnahme erhöht in jedem Fall die Sicherheit des gesamten Programms. Denn auch wenn die Konstanten einmal überschrieben sein sollten, sind sie spätestens zum Lesen des nächsten Satzes wieder regeneriert.

Ist der Wert einer Variablen über eine Kombination mehrerer Operationen zu errechnen, sollte dies nicht in einem einzigen Statement geschehen, sondern zumindest über zwei Zwischenergebnisse. Hier gilt folgender Erfahrungssatz: Ist der Speicherplatz des Endergebnisses überschrieben, so stehen wenigstens die Zwischenergebnisse noch zur Verfügung. Oftmals trifft man auf die Ansicht, daß Ergebnisse nur auf einem bestimmten Wege optimal zu erhalten sind. Übersehen wird dabei, daß der kürzeste oder - wie man sagt - optimale Weg nicht immer der sicherste ist.

Bekanntlich führen viele Algorithmen bei mehrmaligem Gebrauch hintereinander zu unterschiedlichen Ergebnissen. Für den verantwortungsbewußten PPU besteht daher die Notwendigkeit, Ergebnisse mindestens durch zwei verschiedene Algorithmen zu ermitteln. In Abhängigkeit von der Übereinstimmung der Ergebnisse kann auf eine weitere Wiederholung des Rechengangs verzichtet werden, um den Mehraufwand auf ein gesundes Mittelmaß zu beschränken. Bekanntlich sind 50% aller Algorithmen besser als der Durchschnitt (2).

Werden komplizierte Konstanten benötigt, wie z.B. π, so empfiehlt sich aus demselben Grund jeweils eine Neuberechnung (über Winkelfunktionen oder noch besser über eine Reihenentwicklung).

Hiermit nicht im Zusammenhang steht die Forderung, die Läufe auf dem EDV-System mindestens zweimal bearbeiten zu lassen. Stellen sich bei beiden Läufen gleichartige Ergebnisse ein, so kann man die Sache auf sich beruhen lassen. Das zweimalige Bearbeiten desselben Problems empfiehlt sich in Anbetracht des sehr hohen Entwicklungsstandes und der damit einhergehenden Komplexität der heute zur Verfügung stehenden EDV-Anlagen.

Die PPUs werden im allgemeinen nicht in Assemblersprachen programmieren. Gerade für komplexere Probleme bieten sich die höheren Programmiersprachen an, die auch ein größeres Ansehen genießen, wie schon das Adjektiv 'höher' deutlich macht. Sollte aus Gründen, die der Verfasser jedoch nicht akzeptieren würde, ein Programm in einer Assemblersprache geschrieben werden müssen - für diesen Fall soll hier kurz auf folgendes verfahren hingewiesen werden:

Bekanntlich sind Register im Vergleich zum Hauptspeicher sehr schnelle Speicher. Da die Sicherheit mit der Geschwindigkeit nachläßt, empfiehlt es sich, möglichst wenige Registerbefehle und dafür mehr oder ausschließlich Speicher-zu-Speicher-Befehle zu verwenden. Ist einmal ein Registerbefehl unumgänglich, so sollte das Ergebnis umgehend - und sei es auch nur sicherheitshalber - in den Hauptspeicher geschrieben werden. Im übrigen sei auf die Ausführungen über problemorientierte Programmiersprachen verwiesen.

Die PP-Technik im Wechselspiel mit dem EDV-System

Aufgrund der sehr großen Erfahrungen der EDV-Hersteller werden zunehmend Prozeduren bereitgestellt, die genauso gut vom Anwendungsprogrammierer geschrieben werden können. Als Beispiel sei die in den letzten Jahren verstärkt propagierte virtuelle Speichertechnik erwähnt. Sie stellt beileibe keine weltbewegende Prozedur dar. Vielmehr kann sie speziell für jedes Programm oder Programmsystem simuliert werden. Ein derartiger eigenständig entwickelter Algorithmus bietet nicht nur eine selbständige Lösung, sondern kann auch in manchen Fällen die vom Hersteller bereitgestellten Systemprogramme ersetzen.

Im übrigen empfiehlt es sich nicht, das Programm in kleine Einheiten zu zerschlagen und auf peripheren Speichern zwischenzulagern (Paging).

Wenn schon ein Paging notwendig sein sollte, müßte sich der PPU in die Kanalprogrammierung einarbeiten. Das mag als mühselig angesehen werden, es erweitert jedoch die Fähigkeiten des Programmierers.

Die Datenübertragung vom Speicher über den E/A-Rechner, den Kanal und die Steuereinheit zum peripheren Speicher ist leider für den Programmierer transparent. Oben wurde bereits aufgezeigt, daß eigenständig entwickelte Programme oft Systemprogramme ersetzen können. Schließlich sind heute E/A-Rechner, Kanäle und sogar Steuereinheiten programmierbare Geräte.

Um sicherzugehen, daß das Programm von der Jobverwaltung auch bearbeitet wird, empfiehlt es sich, die entsprechenden Systemtabellen zur Laufzeit daraufhin zu untersuchen, ob das Programm auch wirklich läuft!

Einige Betriebssysteme bieten die Möglichkeit des Multiprogrammingbetriebes. Hierbei wird in Hauptspeicherbereichen fester oder variabler Größe gearbeitet. Im Fall der festen Größe der partitions sollte - sofern vorhanden - auf die partition mit größerem Hauptspeicherangebot ausgewichen werden. Die kleineren partitions sind bekanntlich für kleinere Probleme und wohl mehr für Anfänger gedacht.

Steht eine Anlage mit variabler partition-Länge zur Verfügung, sollte der Speicherraum in jedem Fall voll belegt werden, auch wenn eine Nutzung der vollen partition-Länge nicht abzusehen ist.

Obwohl in den meisten Rechenzentren heute mit etikettierten Magnetbändern und Platten gearbeitet wird, sollte darauf hingewirkt werden, daß in Zukunft nur noch unetikettierte bzw. ungelabelte Speichermedien bearbeitet werden. Eine Begründung erübrigt sich für diejenigen, die einmal den Aufwand miterlebt haben, wenn ein Operateur ein falsches Magnetband eingehängt hat, dieses wieder rausnehmen muß, in das Magnetbandarchiv zurücktransportiert, dann das richtige Magnetband herbeiholt und anschließend in das Magnetbandgerät einhängt.

Zum Testen von Programmen sollte in jedem Fall der extended compiler verwendet werden. Er ermöglicht die volle Ausnutzung sämtlicher features des Compilers. Zum endgültigen Lauf empfiehlt sich dann der Run-compiler.

In manchen Compilern sind Ideen verwirklicht, deren Nutzung in jedem Fall ratsam ist. So gibt es z.B. die multitasking-Funktion. Mit ihr ist es möglich, auf Anlagen mit einem Prozessor Anlagen mit mehreren Prozessoren zu simulieren und umgekehrt. Hier können also Ideen, wie das bereits erwähnte Ermitteln von Ergebnissen auf zwei verschiedenen Wegen, leicht verwirklicht werden.

Die meisten Übersetzer werden mit Testhilfen, sogenannten Debuggingtechniques angeboten. Abgesehen davon, daß der Hersteller derartiger Software davon ausgeht, daß Programmsysteme nicht auf Anhieb fehlerfrei erstellt werden können (!), sollte sich der Anwendungsprogrammierer darüber klar sein, daß derartige Hilfen auch in Eigenarbeit programmiert werden können. Äußerstenfalls sollte von dem dump Gebrauch gemacht werden. Es erscheint gerechtfertigt, lieber einen dump zu lesen als sich auf vorgefertigte Fehlermeldungen zu verlassen. Dies gilt insbesondere bei der Benutzung von problemorientierten Programmiersprachen.

Im Wechsel mit der Hardware

Es wurde bereits darauf hingewiesen, daß moderne EDV-Anlagen äußerst komplexe Systeme darstellen. Durch die Installation langsamer Peripheriegeräte ist es möglich, das Gesamtsystem überschaubar zu gestalten. Die Kosten langsamer Peripheriegeräte sind auch geringer als diejenigen der schnellen Geräte.

Bei ihnen läuft der Schreib-Lese-Vorgang entsprechend langsam ab. Es liegt auf der Hand, daß bei geringerer Geschwindigkeit bedeutend besser geschrieben und entsprechend besser gelesen werden kann.

Steht eine Zweiprozessoranlage zur Verfügung, so sollte nur der erste Prozessor verwendet werden, da dieser ohnehin der vorwiegend benutzte ist. Da man heutzutage weiß, daß auch elektronische Bauelemente altern, und wenn man ferner berücksichtigt, daß von zwei Prozessoren jeweils der erste bevorzugt verwendet wird, kann man davon ausgehen, daß der zweite, selten verwendete Prozessor schneller altert (3) und daher früher oder später eine stärkere Fehleranfälligkeit zeigen wird.

Das Design von Programmsystemen sollte so angelegt werden, daß der Hauptspeicher möglichst voll belegt wird. Aus Sicherheitsgründen sollte der gesamte Hauptspeicherinhalt in geringem zeitlichen Abstand auf einen random-Speicher gedumpt werden. Mit diesem Vorgehen erreicht man nicht nur eine hohe Ausnutzung des Hauptspeichers, sondern auch einen hohen E/A-Anteil mit entsprechender Benutzung der Kanäle und Steuereinheiten. Sorgt man dann noch, wie oben erwähnt, für eine hohe Ausnutzung der Rechenwerke durch Mehrfachprogrammierung derselben Algorithmen, so wird man eine hohe Auslastung erreichen. Man spricht dann von einem ausgewogenen System.

Über das Dumpen des gesamten Speicherinhalts hinaus sollten Zwischenergebnisse auf periphere Speicher ausgegeben werden. Selten benötigte Zwischenergebnisse können auf random-Speicher ausgelagert werden, während der PPU oft benötigte Zwischenergebnisse auf sequentielle Speicher ausgibt. Am besten jedes zweite Zwischenergebnis auf ein anderes gerät der gleichen Art. Eine weitere Möglichkeit ist das Ausgeben der Zwischenergebnisse auf verschiedene Platten, die auf demselben Laufwerk angefordert werden. Die Unmutsäußerungen des Plattenoperateurs stellen einen Hinderungsgrund dar, der aber nicht besonders schwerwiegend erscheint.

Oft wird zwischen sequentiellen und random-Speichern unterschieden. Dieses Unterscheidungsmerkmal kennzeichnet jedoch weniger die Geräte als die vom Hersteller angebotenen softwaremäßigen Realisierungen von Zugriffstechniken.

Auch ein Magnetbandgerät kann als Direktzugriffsspeicher programmiert werden. Zum Beispiel läßt sich zwischen den Datenblöcken ein Inhaltsverzeichnis des Magnetbandes abspeichern, zusammen mit Vorwärts- und Rückwärtsverweisen.

Aus Sicherheitsgründen sollte man sich nicht auf ein peripheres Gerät einer Art verlassen, sondern lieber zwei Geräte verwenden. Dies sollte aber nicht dazu führen, daß die Dateien auf verschiedene Geräte gelegt werden. Vielmehr sollte jede Datei gesplittet und dadurch auf jedem peripheren Gerät vertreten sein. Bei gleichzeitiger Benutzung aller Dateien läßt sich dadurch z.B. bei Magnetplattengeräten ein Maximum an Armbewegung erreichen.

Stehen Fernzugriffsgeräte zur Verfügung, so sollte man sie, auch wenn sie nicht zur Eingabe dienen, mindestens zur Ausgabe benutzen. Dabei sollte überlegt werden, ob es nicht doch sinnvoller ist, dieselben Ergebnisse an zwei verschiedenen Orten auszugeben. Schließlich kann nicht immer von einer völligen Ortsunabhängigkeit der erzielten Ergebnisse ausgegangen werden.

Die Kernforderungen der PP-Technik und ihre Ziele

Aus den angeführten Beispielen wird folgendes deutlich: Es geht nicht darum, Software unzuverlässig oder gar fehlerhaft zu gestalten. Vielmehr ist das Ziel der Pessimierungstechnik die Steigerung an Ineffizienz. Anzustreben ist der 7 x 24 Stunden-Betrieb des EDV-Systems. Grundsätzlich gelten die folgenden Sätze:

1) Zu jedem vorhandenen Programm gibt es mindestens ein anderes, das erst noch geschrieben werden muß.
2) Zu jedem Programm gibt es ein anderes, das - 1) vorausgesetzt - den größeren Systemaufwand erfordert.
3) Die Beschaffung des nächst größeren Systems darf nicht länger hinausgezögert werden.

Betrachtet man die Sätze 1 und 2 mehr als Voraussetzungen, so erkennt man in Satz 3 klar die Hauptforderung der PP-Technik. Die Berechtigung dieser Forderung liegt auf der Hand. Diese Forderung läßt sich mit der PP-Technik - ohne langes Nachdenken unterstützen.

Hinweise

(1) Abrahams, P.: Compiler pessimization. Datamation April 1971
(2) Erfahrungssatz des Verfassers
(3) Sprichwort: Wer rastet der rostet