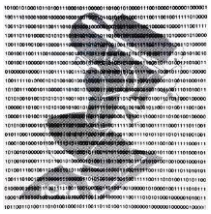
I.C.R.A. – Institute for Computer-assisted Research in Astronomy

Ober Geriwil 41 CH-1715 Alterswil Schweiz E-mail: info@icra.space

Compress Funktionsweise und Verfahren

(vormals Deutsches Patent)



Allgemeines

Zu dem Verfahren "*Compress - Kompression mittels Tokenisierung*" wird Stellung genommen; im Folgenden wird eine grundsätzliche Beschreibung des Verfahrens gegeben.

Ergänzende Literatur stellen die Ausarbeitungen über die Anwendungsmöglichkeiten des *Compress* und *Secure* dar.

Das Verfahren war durch ein deutsches / europäisches und US Patent ab März 1997 geschützt.

Basis des Verfahrens

Die Technologie wird als Software und Hardware, je nach Anwendung eingesetzt; mittels Software kann ein breites Parameter- und Anwendungsspektrum abgedeckt werden, während Hardware aufgrund der fest eingebauten Funktionen einen beschränkten Anwendungsumfang -

dafür meist mit höherer Geschwindigkeit in der Verarbeitung und inhärenter Sicherheit vor Zugriff von aussen - umfasst.

Die Philosophie der Kompression nach *Compress* erfasst langfristige (temporale) und lange (räumliche) Datenstrukturen in kurzen Binärmustern, sog. TOKENS. In der online Datenkommunikation auf synchronen Leitungen läuft ein mehrstufiger Prozess ab, der die langfristige Musterbildung und die zeitlich sehr kurzfristige Musteränderung eines Datenstromes abbildet.

"Token" sind Binärelemente, denen ein Platz (eindeutige logische Zuordnung) eines Wortes oder Strings in einer Frequenztabelle der Muster entspricht, oder allgemeiner ausgedrückt:

> ***Tokens als Binärmuster stellen die kleinstmögliche Darstellung eines Mustervorkommens nach Verarbeitung einer Datenmenge nach Vorkommenshäufigkeiten ihrer Elemente dar.***

In obigen Satz steckt auch die Beschränkung jeder Kompressionsart, die nach einem verlustfreien Verfahren arbeitet: Daten können nur dann komprimiert werden, wenn ihre Entropie kleiner als maximal ist, d.h. die Generierung von Daten mittels eines Zufallsgenerators wird Daten produzieren, die nicht mehr komprimiert werden können. Praktisch finden sich in der menschlichen Kommunikation jedoch keine Zufallsgeneratoren. Das Gegenteil selbst ist der Fall: Rechnerkommunikation läuft mit relativ hoher Redundanz ab.

Das Erarbeiten eines kunden- und applikationsspezifischen Token-Lexikons steht am Anfang der Datenanalyse; dazu wird ein repräsentativer Durchschnitt der zu speichernden oder zu übertragenden Daten auf ihre Token-Fähigkeit untersucht. Das Analyseprogramm ermittelt in mehreren Durchgängen die optimale Verteilung zusammen mit der grösstmöglichen Tokenisierung zeitlicher und räumlicher Muster. Daraus wird das Tokenlexikon der Datenbank oder des zu untersuchenden Datenstromes gebildet. Dieses Lexikon besteht aus dem Muster (als Beispiel ein "Wort") und dem äquivalenten Binärtoken und wird in einer Speichertabelle (feste oder flüchtige Speicher abgelegt). Je nach Anwendung, befindet sich dieses Lexikon auf einer Maschine oder ist verteilt auf die Arbeitsplatzrechner oder wird in Sicherheitsumgebungen jedesmal neu vom Anwender nach seiner Anmeldung von extern (Chipkarte, etc.) geladen.

Anwendung auf Datenbanken

Wird eine relationale **Datenbank mit Token** verarbeitet, so werden sämtliche Suchanfragen über die Indexinformation nach den Speichern verweisen (typisch mittels einer SQL-Abfragesprache), in denen die Information in Form von Token gehalten wird (systemintern natürlich: der Benutzer weiss von dieser Art der Umwandlung seiner Anfrage nichts). In einer Client/Server Umgebung werden diese Token dann von dem Server über ein LAN oder WAN an den Anwender weitergegeben und dort am Ort des Arbeitsplatzrechners in die lesbaren (ASCII -) Symbole umgewandelt. Mehrere Effekte treten als Vorteile auf:

1. die *Suchzeiten durch Kopfbewegungen der Festplatte sind kürzer*, da in einem physischen Element auf der Platte ein sehr viel grösserer Dateninhalt liegt: ***das System wird schneller*;**

2. der *Transport auf dem Netz ist schneller*, d.h. die ***Ausnutzung des Netzes wird sehr viel effizienter,*** da die Token in einem physischen Block auf dem Netz einen höheren logischen Inhalt besitzen;

3. es befindet sich auf dem Server kein Hinweis auf die Originaldaten, d.h. die dazugehörige Lexika-Umwandlung findet erst am Arbeitsplatz statt. Dadurch wird ***eine Art Kodierung der Daten*** erreicht, da sie nur vom Arbeitsplatz aus entziffert werden können;

4. durch die individuelle Erstellung eines Lexikons der Frequenzverteilung mit Zuordnung von Token ist sichergestellt, dass auch bei weiter Verbreitung des ***Compress*** die ***Zugangssicherheit an die Daten eines Unternehmens gewährleistet*** bleibt: im Unternehmen A ist der Begriff "Tisch" an der Stelle 2123 in der Frequenztabelle, beim Unternehmen B steht an der Stelle 2123 "Sicherheitskonzept", usw. Selbst wenn man sich vorstellen könnte, dass man über ein Tokenlexikon verfügt, käme man noch lange nicht an die Daten (da diese natürlich auch durch ihre unterschiedliche Länge und Binärinhalte einen ganz anderen CRC bei einem Blockcheck verursachen).

Anwendung auf online Datenkommunikation

Bei der On-Line **Kommunikation,** insbesondere von synchronen Datenströmen werden bei der Abgabe der Daten aus einer Vorrechnereinheit (Front End Processor) oder einem Router, etc. innerhalb eines Rahmens (“Frame”) an eine externe *Compress* Einheit zu dem Original-String Token durch die Kompression gebildet, ein komprimierter Block aus den Token erzeugt, der Original-CRC des Frame’s eingepackt, ein CRC auf dem komprimierten Frame erzeugt und an das Modem, etc. weitergegeben. Auf der anderen Seite läuft der gleiche Prozess umgekehrt ab. Die Vorteile sind:

1. auf der Kommunikationsleitung beansprucht der Datenverkehr weniger Zeit: eine ***bestehende Leitung kann ca. um den Kompressionsfaktor höher betrieben werden*;**

2. durch das Synchronisationsprotokoll der beiden *TEC-Compress* Einheiten wird erreicht, dass ***nur mit einem Compress Teilnehmer, und nicht mit einem fremden Teilnehmer Verbindung aufgenommen werden kann*** (unbefugtes Anschalten, Authentisierung);

3. dem Token - Datenstrom kann ein ***Chiffrierschlüssel*** **mit einer Million Bit Länge** aufgeschaltet werden, der nur mittels der anderen ENCRYPTOR Einheit

wieder zu den Tokens und damit zu den Daten führt; damit wird eine hochsichere Datenübertragung gewährleistet;

4. *Compress* ist ***transparent zu einer Anwendung***; Protokolle unter der Transportebene (IP, TCP, etc.) werden nicht berührt. Das Transportprotokoll wickelt den richtigen Erhalt eines Frame ab.

Das Beibehalten des Original-CRC bildet eine doppelte Sicherheit: der komprimierte Frame wurde als solcher richtig dekodiert durch den CRC des komprimierten Frames; der Original-CRC bildet die untrügliche Information über die Rückbildung des unkomprimierten Frame-Inhaltes.

Hardware versus Software

ICRA hatte bereits seit 1985 einen integrierten Schaltkreis als VLSI in HCMOS Technologie im Einsatz, der in Zusammenarbeit mit einer CPU aus der iNTEL - Prozessorfamilie die Funktion eines sehr schnellen Prozessors für Mustererkennung ausübte. Die Einheit, eine Karte mit Logik und Prozessor, ist mit Speichern (als ROM, SRAM oder RAM) versehen, in denen die sogenannten 'Token' abgelegt sind. Heutige Anwendungen können in Software erfolgen, ohne dass Rückgriff auf eine spezielle Hardware genommen werden muss. Sicherheitsanwendungen können auch heute noch die Hardware erfordern; ein moderner Prozessor steht zur Verfügung.

Compress Vorgehensmodell

Man ermittelt aus einem Text- (oder Binärelementen-) Raum Muster, die in einer Frequenztabelle mittels der von ICRA entwickelten Analyseprogramme abgelegt werden; meist beschränkt sich eine solche Tokenisierung auf die ganzen Wortelemente einer Sprache, bzw. Zahlenelemente in einer Datenbanktabelle oder Applikation (Druckaufbereitung wird meist seitenweise verarbeitet). Dabei werden den häufig vorkommenden Termen kurze Token und den weniger häufig vorkommenden Termen längere Token zugeordnet. Es können ganze Worte, Teile davon (auch als *'n-grams'* in der Literatur bezeichnet) und auch andere beliebige Muster als Token assoziiert werden ("Wort", aber auch "Mit freundlichen Grüssen" kann ein Token sein, sowie auch "...ion ab..." und "...es ef...", sowie ganze Sätze in sehr strukturierten Umgebungen).

> *Beispiel 1:*
> In einem bestimmten Sprachlexikon eines Unternehmens sei das Wort 'TISCH' in der Frequenztabelle an Stelle 678 angeordnet. Der dazugehörige Token ist das Binär-Äquivalent von 678, also eine 10-Bit Kombination von 0 und 1 (512 bis 1024 oder 2^9 bis 2^{10}).
>
> Im Raum 2^{10} kann die Zahl 678 eineindeutig dargestellt werden. Der ursprüngliche Weg der Speicherung und Übertragung benötigt bei einem 8-bit

ASCII Code 5x 8 Bit (= 40 Bits), in der ***Compress*** Version 10 Bit; also ein Faktor 1 : 4 an Speicher oder Übertragungsmenge.

Beispiel 2:
In einem Datensatz kommen zusammengehörige Elemente vor, die mit der Struktur der Datenbank in Zusammenhang stehen, im einfachsten Falle die gesamte Adresse oder der gesamte Name eines Kunden. Obwohl eine Tokenisierung der Elemente *Heinrich* und *Müller* eventuell bereits vorliegt, wird durch einen zweiten Durchgang der Analysesoftware ein Token aus den beiden Elementen *Heinrich (Leerzeichen) Müller* gebildet. Nun ist die Ersparnis nicht mehr nur die Summe der beiden Token (Heinrich => 64 Bit, Müller => 48 Bit, Space => 8 Bit, oder ohne Tokenisierung: 120 Bit), bei einer Frequenzhäufigkeit vom 1000. Platz in der Tabelle (10 Bit-Token) nicht mehr zwei Token von 10 und 10 Bit, sondern **nur ein** Token und auch nur 10 Bit, oder eine Kompression bezogen auf den Roh-Bitwert von 1:12.

Den Token kann man auf verschiedene Arten behandeln. Im geläufigen Sinn würde man die 10 Bit auf 16 Bit mit 000000 auffüllen (Padding), um ein für den Prozessor komplettes '16-Bit Wort' zu erhalten. Wenn man aber Token bis zu einer festgelegten Blocklänge (128 Bit, 512 Bit, 1024 oder grösser) aneinanderreiht, ohne jeden Token auf 16 Bit anzufüllen, erreicht man eine bessere Kompression und einen interessanten Nebeneffekt:

Durch das Aneinanderreihen der Token und nachgeschaltetes Zerteilen in 16 Bit Elemente durch den Prozessor (damit der Prozessor mit einer Bitfolge arbeiten kann, muss er sie in seine 'Wortlängen' trennen; bei der CPU i286 sind dies 16 Bit, bei der Familie i386 und i486 sind dies 32 Bit) verschwindet die Information über Anfang und Ende eines Token, da ein Token z. B. 4 Bit im ersten CPU-Wort haben kann, durch die Aufteilung in die 16 Bit Wortlänge dann die folgenden Bit eines 9 Bit Tokens im nächsten CPU-Wort erscheinen.

Der Effekt ist die totale Unkenntlichkeit von Tokenverteilungen, also *eine Art Kodierung,* da die *Verteilungsfrequenz* der Tokens nicht mehr ersichtlich ist.

Beispiel:
'Wort' bezeichnet ein 16 Bit Wort des Prozessors.

```
Token 1       Token 2       Token 3           Token 4   Token 5              Leer
000110011|101101110|000110001110|1010100|111000101010001|00000
 (9 Bit)    (9 Bit)    (12 Bit)      (7 Bit) (15 Bit)

.............|.............|.............|.............
 1. Wort       2. Wort         3. Wort       4. Wort
```

CPU Worte (16 Bit) eines Computers.

Es kommt bei dem ***Compress*** - Verfahren darauf an, während der erststufigen Datenanalyse die langfristigen Tokens zu erkennen und die kurzzeitigen Änderungen einer zweiten Kompressionsstruktur in Form eines Baumes zuzuführen. Da die langfristigen Token nach

Frequenz orientiert sind, werden sie häufig kleine Token in die Baumstruktur liefern, sodass diese klein bleiben kann.

Nicht alle Datenstrukturen eignen sich optimal für Tokenbildung. Um in einem Unternehmen den wechselnden Anforderungen gerecht zu werden, ist ***Compress*** mit einem mehrstufigen Verfahren versehen: das Mass der Kompression wird dynamisch bei jedem transferierten Block gemessen und mit den letzten Werten verglichen; ändert sich das Kompressionsverhalten signifikant, schaltet die Maschine automatisch auf kurzzeitige Verfahren um, behält jedoch die langfristigen gelernten Strings. Dies gilt besonders für schnell wechselnde Datenarten. Durch eine vorherige Analyse des Kommunikationsaufkommens sollte ein solcher Wechsel weitgehend vermieden werden können.

Aus Obigem wird auch ersichtlich, dass das Verfahren weniger für sporadisch wechselnde und nicht-analysierte Datenarten geeignet ist. In heutigen Unternehmensumgebungen jedoch findet man sehr stark strukturierte Daten vor, die entweder zur Speicherung kommen, oder zwischen mehreren Stellen kommuniziert werden.

Verschlüsselung - *Secure*

Compress lässt sich für Verschlüsselung verwenden: will man eine Übertragung oder Speicherung verschlüsseln, sieht ein einfaches Verfahren (ein-stufiges Cryptosystem) folgendes vor (andere Verfahren mit Public Key Anwendungen, wie RSA, IDEA, etc. können in Absprache mit dem Kunden implementiert werden):

Die Schlüsselgenerator Software erstellt man mit Hilfe eines Zufallsgenerators (Eingangs-parameter oder *Seeds*: RAM Zustände des Computers, Zeit in Millisekunden, eine komplexe Funktion, etc.) einen Schlüssel mit grosser Länge.

Warum einen Schlüssel mit grosser Länge ?

Die Wiederholung eines Codes, z.B. mit einer Million Bit Länge erfolgt erst nach einer Million Bit oder nach ca. 132 KByte. Dies bedeutet, die Entschlüsselungswahrscheinlichkeit liegt bei $2^{1'000'000}$, eine sehr grosse Zahl selbst für Supercomputer, gegenüber den 56 Bit eines konventionellen Schlüssels wie DES.

Dieser *Encryptor* - Schlüssel (für jeden Benutzer individuell erstellbar und in beliebiger Anzahl) wird den Token aufgeschaltet, z.B. werden alle 8 Bit der Token mit dem Schlüssel via eines logischen XOR versehen und das resultierende Byte (8 Bit Wort) an die CPU zur Entsendung, sei es auf Platte, LAN oder Modem, etc. gegeben. In der obigen Darstellung muss man also das erste Byte des Schlüssels mit dem ersten Halb-Wort oder Byte des(r) Token mit XOR versehen und erhält das resultierende endgültige Byte im Computer. Um es noch schwieriger zu machen, gibt man dem Schlüssel einen sog. Offset mit, d.h. man legt fest, dass das erste Byte des Schlüssels z.B. mit dem Bit 234543 des Schlüssels begonnen werden soll und nach einer auf beiden Seiten abgelegten Formel verfahren werden soll.

Die Vorteile sind, dass man nur mit dem Schlüssel **und** der *Compress* Hardware **und** dem Lexikon die Originaldaten wiederherstellen kann. Wenn man den Schlüssel auf einer optischen Speicherkarte hält (Scheckkartengrösse), ist jedem Benutzer der Zugang an ein System individuell anpassbar und jeder Mitarbeiter kann an jedem Arbeitsplatzrechner an dem System arbeiten.

Zukunft von Datenbanken

Es wird in Zukunft in der Datenspeicherung und Übertragung ein Umdenken stattfinden müssen, da man nicht mittels Passwörtern und ähnlichen Methoden Hacker am Eindringen in Datenbanken und dem Verwenden dieser auf Satellitenleitungen und Telefonleitungen transportierten Bits Einhalt gebieten kann. Mit einer Einrichtung für wenig Geld kann man eine Übertragung abhören, wenn die Daten über eine Satellitenleitung übermittelt werden.

Der Trend muss zu *kodierten Daten in Datenbanken* und auf Kommunikationsleitungen gehen, die nur mittels einer Hardware **und** einem dazugehörigen Schlüssel gelesen werden können und deren Sender sich authentisieren lassen (*Message Digest* Verfahren oder andere). Zumindest sind dann die Personen bekannt, die eine solche Hardware besitzen. Dazu bleibt sicherlich weiterhin der Passwort-Zugang an die Maschine erhalten, hat jedoch keinen Einfluss auf die Sicherheit einer Datenübertragung mehr.

Geschwindigkeit der On-line Verschlüsselung

Da die *Compress* online Einheiten heute mit bis zu ca. 8 Millionen Bit in der Sekunde arbeiten, fällt die Arbeit der Verschlüsselung nicht ins Gewicht, d.h. es gibt keine merkliche Verzögerungen für den Benutzer durch die Verschlüsselung.

> *Beispiel:*
> Eine 40-seitige Dokumentation (ca. 80 Kbyte) wird mit Verschlüsselung in ca. 5.38 Sekunden parallel verarbeitet, ohne Verschlüsselung in 5.18 Sekunden.
>
> *'Verarbeitet'* heisst:
> - On-line komprimiert und verschlüsselt im sendenden *Compress*,
> - parallel versandt in der *Compress* Einheit,
> - dekomprimiert und entschlüsselt im *Compress* -Zielsystem.

Man kann mit den Produkten *TEC-Compress* (1999) bis zu 512 Kbit/Sek. online verarbeiten in Abhängigkeit von der Anzahl der Frames. Bei diesen Geschwindigkeiten können mehrere z.B. 192 Kbit Leitungen online komprimiert werden, verschlüsselt und online entschlüsselt und de-komprimiert werden.

Anwendungen auf geschriebene Sprache

Jede beliebige Binärmenge (unterhalb des maximalen Entropie-Wertes) kann durch Muster dargestellt werden; statt der Originaldatenmenge werden kürzere Token versandt, mit denen im Rechner gearbeitet wird. Eine solche Anwendung ist die Sprachrepräsentation in Worten in ASCII Zeichen.

Für die englische Sprache hat ICRA bereits 1985 ein *allgemeines Lexikon* auf der Basis des "American Heritage Frequency Word Dictionary" erzeugt. Man erzielt unter Verwendung der ersten 28'000 Wörter in der Frequenztabelle eine gemittelte Kompressionsrate von ca. 1:3. Im zweistufigen Verfahren (Ermittlung der langfristigen und kurzfristigen Muster) werden Kompressionsraten von ca. 1:5 erreicht. Solche Versuche dienen als eine quasi-"worst-case" Ermittlung, da die bekannten Token nichts mit den speziellen Daten gemeinsam haben. Aus dem vorher Gesagten ist auch ersichtlich, warum eine individuelle Anpassung, d.h. Erstellung eines eigenen Token-Lexikons im Unternehmen - neben dem Sicherheitsaspekt - für den Kunden bessere Ergebnisse liefert.

Spezielle Anwendungen

Werden die Programme zur Erzeugung eines speziellen Satzes von Lexika für einen bestimmten Wortraum firmenspezifisch verwendet, liegt der Kompressionsfaktor deutlich höher, da die gewählte Umgebung meist aus sehr hochfrequenten Elementen besteht.

Als Beispiel sei die Darstellung einer grossen Datenbank innerhalb einer internationalen Bank genannt, wo der hochfrequente Sprachraum ca. 8000 Terme umfasst und deshalb eine Kompression von ca. 1:17 als Resultat über mehrere Typen von Dokumenten, und Datenbanken erreicht wurde.

Die Auswirkung einer solch hohen Kompressionsrate bringt Ersparnisse im Bereich Plattenkapazität für Datenbanken - es wird nur noch ein Bruchteil (hier: ca. 1/17) der Speichermedien benötigt. Die Kostenersparnis ist enorm. Dies wirkt sich jedoch auch auf die Geschwindigkeit einer Suche aus: die Plattenkopf - Zugriffe sind weniger und die damit verbundenen Wege auf der Platte geringer. *Die I/O Operationen sind wesentlich geringer*.

Weiterhin ergibt sich durch das Verwenden von Token eine bedeutend schnellere Antwortzeit für den Benutzer und die Effizienz des lokalen Netzes wird enorm erhöht, da die Token mehr logischen Inhalt besitzen als zuvor, und erst am Arbeitsplatz entschlüsselt und de-komprimiert werden.

Vorteile und Beschränkungen

Der Hauptvorteil von sog. statischen Verfahren (wie Tokenisierung) liegt in ihrer Verwendbarkeit in einem breiten Anwendungsspektrum, welches bis hin zur Volltextdatenbank (siehe *IMS*) reicht.

Gegenüber adaptiven Verfahren (Ziv-Lempel-Welch), wie sie heute in den V.42bis Modems, etc. Eingang finden, ist *Compress* nicht auf kleine Speicher angewiesen. Die Kompressionsrate ist damit weniger von der Länge einer Datei oder eines Rahmens (Frame) abhängig. Somit kann auch in interaktiven online Verbindungen immer mit einem guten Kompressionsfaktor gerechnet werden, auch wenn immer nur kleine Mengen Daten versandt werden (typische Anwendung: Reservierungen bei Airlines - mittlere Übertragungsmenge ca. 150 Byte).

Das Halten eines Lexikons ist die kritische Frage in den adaptiven Algorithmen - ab welchem Füllgrad eines Speichers wann wirft man das Gelernte weg und beginnt von neuem? Damit werden meist nur die kurzfristigen Elemente verwertbar, langfristige Vorkommnisse werden nicht erfasst.

Der Sicherheitsaspekt der adaptiven Verfahren ist ebenfalls interessant: adaptive Verfahren sind selbsterklärend (‘an die Stelle 12 des Baumes setze ein “F“ ’ - nachdem der Baum vorher auf die gleiche Vorgehensweise aufgebaut wurde). Sie eignen sich also nicht für die Übertragung von sicherheitsrelevanten Daten.

Die Logik des *Compress* lässt ein Lernverhalten der Maschine zu: wird ein Wort nicht erkannt, so kann es als Token ausgedrückt für spätere Wiederholung komprimiert versandt werden. Die Tokentabelle wird dabei dynamisch um eins inkrementiert.

Wird Compress mit bereits komprimierten Daten versehen, wird die Maschine je nach Entropie der Daten versuchen, weitere Mustergruppen zu bilden. Es ist jedoch aus der Theorie ersichtlich, dass eine bereits komprimierte Datei oder ein Datenblock mit einem Höchstmass an Entropie auch von *Compress* nicht weiter komprimiert werden kann; in diesem Falle übermittelt *Compress* die Daten ohne Veränderung, um sicher zu stellen, dass kein "Aufblähen" des Übertragungsvolumens eintritt.

Inhalt

ALLGEMEINES **1**

BASIS DES VERFAHRENS **1**

ANWENDUNG AUF DATENBANKEN **2**

ANWENDUNG AUF ONLINE DATENKOMMUNIKATION **3**

HARDWARE VERSUS SOFTWARE **4**

***TEC-COMPRESS* VORGEHENSMODELL** **4**

VERSCHLÜSSELUNG - *TEC-SECURE* **6**

ZUKUNFT VON DATENBANKEN **7**

GESCHWINDIGKEIT DER ON-LINE VERSCHLÜSSELUNG **7**

ANWENDUNGEN AUF GESCHRIEBENE SPRACHE **8**

SPEZIELLE ANWENDUNGEN **8**

VORTEILE UND BESCHRÄNKUNGEN **9**

INHALT **10**