45 9

タイムシェアリングの原理と応用

スタンフォード大学 教授

工学博士 JOHN McCARTHY

昭和44年3月

財団法人 日本情報処理開発センター

NPDEC

本印刷物は、当財団中央研修所の特別講師として招聘したスタンフォード大学　マッカーシィ教授の一般講演会の内容を、通訳者の速記録から編集のうえ作成したものであり、文責は当財団にある。

タイムシェアリングの原理と応用

スタンフォード大学教授　工学博士　JOHN McCARTHY

コンピュータのタイムシェアリング利用についてビジネスにおけるコンピュータの応用とか、科学計算といつたような、タイムシェアリング以前の応用に関しても触れながら述べてみる。

まず、コンピュータに何を行なわせるかを考えてみると、コンピュータが理解できる形でデータを与えてやり、問題を正確なアルゴリズムで指定しさえすれば、どんなことでも出来るといつてよい。　正確なアルゴリズムの指定ということは、プログラミングの本質に関係する問題であるため1つの例をあげて話を進めることにする。

図　1

```
begin  read (N) ;
       n ;= 0 ;
  a :  n×n   > N  then go to done ;
  n :  =  n + 1 ;
  go   to   a ;
  done  :  type  (n)  end ;
```

図1に示すように、ある数の平方根よりも小さな数でその内で最も大きいものを求めるという計算の手続を正確に定義してみると、完全平方数に対してはその平方根が計算され、その他の場合には平方根に最も近い数が計算される。

図1に示したプログラムは、このために正確にアルゴリズム化されたものであり、これをカードにパンチしたり、あるいはタイムシェアリングのターミナルにタイプするといつた方法で、Aという数の平方根に最も近い数を求めることができる。

このプログラムでは、まずnという字で表わされる量をゼロに設定し、nとnの積をとり、それがNより大きいかどうかを設定する。　まだNより小さいときには、nに1を加え（n＋1）×（n＋1）とし、それがさらにNよりも小さければ、次の段階へと繰り返し演算を行なう。

たとえば、ある数を２乗したものが４より大きいかを比べる場合１と１の積をとり４より小さければ１に１を加え、２についてやつてみる。　このような操作を繰り返して、２乗が４よりも小さい数で、最も大きな数を求める。

この例の主眼点は、正確なアルゴリズムで問題を書くことであつて、コンピュータが受けとることができるような形で問題を記述して初めて、コンピュータを使用できるということである。

今日、コンピュータの使用方法として、バッチという方法があるが、この方法では、問題をカードにパンチして、上の例のようなランゲイジあるいはその他のコンピュータ用ランゲイジで問題を書きデータを添えてデータセンターに持つていくわけである。　この場合、いろいろな人がデータを持ち込むため、待ち行列ができるが、自分の番がくるまで待つて、問題を解いてもらう形をとつている。

計算センターで、バッチ処理を行なつている場合は１日に１,０００ぐらいの仕事をやるのが普通であつて、中には４,０００とか５,０００とかいつた仕事を片づけているところもある。

しかし、コンピュータをこのように使うのが適当でない場合があり、ここでその例を２，３あげてみる。

(1)　コンピュータを学校の先生の代わりに使用する場合で、これは過去数年来、主に米国で実施されているが、コンピュータを生徒の個人教授に使用しようとするものである。

この場合、生徒に何か情報を与え、それに関してコンピュータが質問をし、生徒に答えさせる方法をとる。　コンピュータはその答えを評価し、次にどういう質問を出すかまたはどういう情報を与えるかということをきめていくわけである。

(2)　制御装置に関して私どもの研究所では、人間の手と同じ働ができる機械手を作成し、コンピュータでコントロールする研究を行なつているが、さらに機械とテレビジョンカメラを使い、目と組み合わせていろいろなことを行なわせてみる研究も行なつている。

さらに私達の研究所では、乗物をコンピュータで制御しようとする研究を行なつている。　たとえば、自動車をコンピュータで制御しようとするものであるが、これは今まで全く異なつた方法をとらなければならない。

もう１つの例は、簡単な計算をコンピュータを利用して行なう場合である。

今日のコンピュータの速さは、人間の計算スピードの１００万倍とか、あるいはそれ以上のものであつて、たとえば掛け算を１０００回行なう場合、コンピュータを使用すると1秒の何千分の1といつた単位で行なうことができる。

もちろん、今日の高速のコンピュータでも非常に時間のかかる、いわゆる重い計算というものもあり、今日の最も速いコンピュータでも数時間を要する場合もあるが、コンピュータはますます速くなりつつあり、演算スピードが上がるにつれて上に述べた簡単な計算の分野というものはどんどん広がつていくように思われるが、常に、できるだけ早く解答を得たいことになる。

1回きりの計算だけで終了せず、ある計算の結果をみて、次の問題を考え、その問題について演算を行ないたいといつた逐次的に問題解決を行ないたい場合、コンピュータ自身のスピードというよりは、何時でもコンピュータを使用することが要求され、これがタイムシェアリング発達の大きな理由となつている。

タイムシェアリングにおける演算では、インストラクション1つに対して1マイクロ・セカンドが標準と考えられる。　タイムシェアリングが実施された当初は、これよりもかなりおそいスピードのコンピュータで行なわれたが、今後、次第に演算速度が速くなることで話を進めていくことにする。　図1に示した問題を、タイムシェアリング用コンピュータで行なう場合は、ユーザーがターミナルのタイプライターに何かタイプし、コンピュータにNの特定の値に関し演算させることになる。

非常に速いタイピストであれば、大体1秒間に5字タイプすることができるが、普通のタイムシェア・ユーザーでは、平均3秒間に1つぐらいの割合でしか字を打てない。このことは、ターミナルを使つている間、考えている時間もあれば、計算機からの出力を受けとつている時もあることから言える。

情報をコンピュータに与えると、ユーザーは、エンドをタイプしなければならない。これによりコンピュータは何をするか判断する訳で、使用者からの情報がこれでおしまいだということをコンピュータが知らないと、次にどういう情報が入つてくるかを待たなければならず、受信済みの文字を記憶装置に入れねばならない。　そういつた1つ1つの文字が何を意味しているかを調べる時間はコンピュータによつて異なるが、１０マイクロ・セカンドから３００マイクロ・セカンドといつた時間と考えてよい。

ユーザーが、3秒たりないと次の文字をタイプしないといつた場合は、最も遅いコンピュータを考えてみても、いま入つてきた文字が何であるかというのを調べるのに、わずか自分の時間の1万分の1しか使つていないわけであつて、1万のうち９９９９という時間

は、待時間で休んでいるわけである。　これは、コンピュータにとつては非常に無駄な使用の仕方であつて、待つている間はかの人の仕事をさせた方が良いことになる。

あるユーザーから入つてきた文字に関して、何か不都合なことがあつた場合は、コンピュータは、そのための処理を行ない、次の文字が同じ使用者から与えられてくる前に仕末しなければならない。

というのは、タイプライターからは１度に１字しか送れないのでコンピュータは、その１字を摑まえなければならないからである。

使用者がタイプライターの２つのキーを非常に早く相次いでたたいた場合を考えてみると、１０分の１秒以内にユーザーに対して何かしなければならないことにもなる。

こういつたことから、コンピュータに割り込みをかけて、いまやつていることを中断させて、迅速にその誤りに対する方策を立てさせることが何らかの形で必要になるわけである。

コンピュータがこういう処置を取るのに必要な時間は短いが、その処置は早く始めないと問題の字は失われてしまうことがある。

こういつた場合の処置として、２つの方法があり、普通の方法は、何れかのタイプライターから文字が入つてくるたびにコンピュータに割り込みをかける方法である。

コンピュータは、この場合、自分がどこに送つたか、何をしていたかということを覚えておいて、処置するわけで、本質的には、次の命令のアドレスを覚えておいて、そのプログラムにもどるわけである。　そのプログラムを見ると、どこのタイプライターがいまひつかかつたか、あるいは、故障かというようなことがわかるようになつている。

もう１つの方法は、時計をつけておき、コンピュータに時間的に割り込みをかける方法である。　これは、たとえば0.1秒に１回であるとか、１秒に１００回であるとかいつた割合で演算を中断し、いずれかのタイプライターからいま何か文字が入つてきているかどうか、入つてきているとすれば何をするべきかをコンピュータに決定させる方法である。その他、コンピュータがあるタイプライターに何か通信文を印刷している過程を考えた場合、１秒間に１０文字タイプする程度であるが、この場合そのタイプライターがいま文字を受け入れる状態になつているかどうかを調べる必要がある。　タイプは、ユーザーが３秒に１つずつ文字を打つ可能性があるので、いまコンピュータから文字を送つていいかどうかを調べる必要があり、また、まだ出力すべき文字があるかどうかも見る必要がある。

さらに、コンピュータは次の事も仕事をやる前にきめなければならない。　つまり、あるユーザーが何か計算をしたいといつてきている場合に、その計算のため使用時間を全部

使つてしまつたかどうか調べなければならない。　使用時間がすでになくなつている場合は、その他の希望者の仕事を始めることになる。　これは、ある特定の人が長い計算を1度にやつてくれとコンピュータに要求し、それを許すことにすると、他の人の仕事は、その間全然できないことになる。　このため、タイムシェアリングでは、使用者に対し、1度に使用できる時間を制限している。　この決められた時間のことをクオンタイムという。

ここで図1に話を戻すと最初に　begin と書き、Nの値を読む。　その後、答のnが幾らかということをタイプしてほしいという要求をしている。

このプログラムが終わつた後、ユーザーが、ファイルしたいとコンピュータに指令し、結果をコンピュータにストアしておくと将来このプログラムを使用したいときには、ユーザーは、run Root といつて同じプログラムを実行することができる。

ユーザーがrun Root をコンピュータに知らせ、プログラム中の Reed Nの所にくれば、コンピュータの方からは、Nとは一体何のことだとか、Nの値を指定して下さいといつた適当な形のメッセージが出て、使用者がたとえば、Nの値として25といつた数字をタイプすることになる。

このように計算が続行され、最後にコンピュータの側から5という値をユーザーにタイプして戻すことになる。

run Root の場合、幾ら位の計算量が実際に必要かということは、Nの値を幾つに指定するかということによつて決まり、Nが4であると計算は非常に簡単であるけれど、Nがたとえば100億であるとすると、計算量は非常に大きなものになる。

このように、プログラムが書かれた段階では計算量は不明であり、それは指定するデータによるわけである。

タイムシェアリングシステムがこういつた問題を処理するのに、どのような動作をしなければならないかをここで触れてみる。

まず、コマンドが来た時、コンピュータの方からは、たとえばNの値をタイプしてよこせといつたようなメッセージを出すことになる。

その後、コンピュータはユーザーからのコマンドを解釈し、コンピュータにtype in Nを印刷させよと指令がくれば、印字することになる。
この場合は、タイプライターのスピードが1秒あたり10文字といつたことから、メッセージが出終わるまでには、1秒位の時間がかかることになる。
しかし、このプログラムの使用者は、コンピュータからのメッセージに応答して平方根を求めるべき数をタイプしてくれる速さは分らない。

したがつて、１０秒くらいで応答してくれるかもしれないし、1時間かかることもあり得る。

実際には、type in N というコマンドは記憶装置の中の出力バッファと呼ばれる特定の領域に一たん入れておき、そこからユーザーに対してNの値は、と問い合わせることになる。　この場合、タイムシェアリングシステムにユーザーが何かいつてきたら行動を起こせとコンピュータ側から指令する訳だが、処理が高速度（１００マイクロ・セカンドもあれば充分）行なわれるためユーザーからの応答が間に合わない場合もある。
この場合、一たんその仕事を中止し、他のユーザーの仕事にとりかかることになるが、ここで注意しなければならないのは、ユーザーからのプログラムをすでに１文字を読み取ろうとしていることである。

すなわち、他のユーザーの仕事を１０秒とか、１秒とか１分とすれば、その後このユーザーが要求された数字を打ち込むと、直ちにタイムシェアリングシステムはこの数字をプログラム中のNと置き換える。　次にシステムが数字の終了を理解する方法として次の２つの方法が考えられる。　というのは、３という数字を考えた場合、終りの印がなければ３５２５といつた一連の数字の最初の数字を示しているかも知れないからである。

(1)　数字を読込んで後、読み込みを終了させて演算に移る。

(2)　数字の次に異なつた英文字がくれば、その英文字までを数字として取扱えとシステムに指令しておく。　したがつて異なつた文字がくるまで読込みを続ける。

最初の方法では、ユーザーは、３という数字を打つてプログラムが受け取つた後、次の字は何だという要求に従つてユーザーは４を打ち、次に５を打つというように繰り返し、最後にユーザーがスペースを打つたときに、これは数字の終りだということを判定することになつている。

２番目の方法は、ユーザーが数字でないキャラクターを打つまでは、プログラムはその数字を受け取らず最後にスペースを受け取つて後、演算を開始する。コンピュータがこの数字を受け取ると内部コードに変換して計算し、解答をタイプライターから打ち出すことになる。

この場合、与えられたクオンタイムのうちにその計算が終了してしまうか、計算が終る前にクオンタイムが終わつて、誰か他の人が割り込み、後に次のクオンタイムをもらい、仕事を継続するという２つの場合が考えられる。

上に述べた２つの場合で、与えられたクオンタイムのうちに仕事が終つてしまう場合は、ユーザーはきわめて速くサービスを得ることができる。　ユーザーが簡単な計算を行ない

たい場合、瞬間的なサービスを提供するという印象をユーザーに与えるためには、0.2秒という時間にする必要がある。　というのは、0.2秒というのは、人間の反応する時間であつて、たとえば、灯がついたときにボタンを押せといつた場合、非常に応答の速い人ならば、灯がついてから0.2秒内にボタンを押せることからきている。

次にアクティブユーザーとは、

実際に、コンピュータに何かさせたいと思つている人の数のことである。　自分のプログラムに自分の打鍵等を待たせているユーザーやアウトプットが出されつつあるユーザーはこの数の中には入らない。　アクティブユーザーの数が例えば１０名であるとすると、クオンタイムは５０分の１秒にするべきであるという結論が出てくる。

しかし、これには若干の問題があつて、すべてのアクティブユーザーのプログラムがコアメモリーの中に入つている場合はよいが、そうでない場合は時間がかかることになる。

すなわち、コアメモリーであれば１マイクロ・セカンドとかいつた時間の間にプログラムの読み出しができる。　したがつて、システム内で特定のユーザーの時間がすでになくなつてしまつたかどうかを調べて他のユーザーの仕事に切り換えるのに高々数マイクロ・セカンドないし数十マイクロ・セカンドの時間があればよいため、短いクオンタイムを割り当てることは非常に有効である。

最近やつとコアメモリーの値段がユーザーのプログラムが全部コアに入つているようなタイムシェアリング・システムを構成できる程、下つてきたが、それでも私は、こういう使い方をしたタイムシェアリングの実例はただ１つしか知らない。　普通は、プログラムが実行されていない時は、何らかの２次メモリーの中に格納しておくことになる。　このメモリーは、実際にコンピュータが動作する時に使うメモリーよりも安いが、スピードが遅いのと、２次記憶装置に入つているプログラムをコアメモリーにもつてくる場合、時間がかかることに難点がある。

また、すでにコアメモリーの中にある他のユーザーのプログラムを外へ追い出すとして、場所をあけなければならないといつたことが生ずる場合があり、それにも時間がかかる。ユーザーのプログラムをコアメモリーの中に入れる場合、コアの中で他のプログラムを実行できる場合もあるが、待たなければならない場合もあり、こういつたことから、オーバーヘッドといつた問題が生じてくる。　クオンタイムをあまりに短くするとオーバーヘッドが決定的なものになつてしまい、コンピュータは時間を浪費し、ある種のタイムシェアリングシステムでは、これが重要な問題となつている。

どのようにクオンタイムを決めるかについては種々要素があるが、プログラムの平均の

サイズであるとか、２次メモリーと主メモリーとの間の転送速度といつたもの、さらに２次メモリーがドラムであるとかディスクであるとかによるアクセスタイムが問題になつてくる。　ここで、これに対して最適と思われるシステムの１例を上げてみる。　ユーザーのプログラムが全部コアの中に入つている場合を除くと、大体4,０００ワード毎にプログラムをスワップアウトやスワップインするのが良いと考えられる。　この場合に、ラウンド・ロビンタイムがどのぐらいになるかは、計算のロードがどのぐらいの重さかに関係するが、１０分の１秒から1.５秒といつたことになる。

次に、コンピュータを教育に利用することに関して、もう少し詳しくお話しする。
現在までに、私がいるスタンフォード大学では、PATRICK SUPPES 教授がこの分野でのプロジェクトをもつているが、そこで現在行なわれつつあること、さらに今まで行なわれたことに重点を置いて述べてみる。

小学校で算数の練習に使つている例を上げると、教室にテレタイプを置き、子供が毎日５分とか１０分とか、教室でテレタイプを使つてコンピュータに算数の練習の相手をしてもらうわけである。　この方法は、たとえば３＋４といつたような問題を生徒に出し生徒たちが時間内に７といつた答を出すと、それを見てその生徒が練習を終えるまで次の問題を出し続けていくことになつている。
いまのは早く応答した場合であるが、応答がおくれたり、全然ボタンを押さない場合にはコンピュータはその生徒に時間切れであることを伝えてもう一度問題を出すことになる。それでも駄目で、全然できない場合には、答を教えて次の問題を出し間違つた答を出した場合には、間違いを生徒に教える。　誤りを、２度繰り返すと正しい答を生徒に教えることになる。

このプログラムは、実際に毎日1,５００人の生徒に対して問題をやらせているばかりでなく、生徒の成績のレポートを作成することもできる。　どのようにその問題を理解しつつあるかが分かるようになつていて、１人１人の先生が教えるよりすぐれていると考えられる点が多い。

これと同じ方法で、教育面への応用は、コンピュータで言葉を発生させることである。これは言葉がディジタルな形でディスクファイルに入つていて、電話線を介してユーザーの所へ届いて再生されるわけだが単語のスペルのつづり方を教える科目で用いられている。

もう少し野心的なことにも使われている。　たとえば、スタンフォード大学では、４年目以下の学生にロシア語を教えるのに、このシステムが使用されている。　すなわち、５つのテレタイプライターがあつて、タイプライターで、ロシア文字とラテン文字を取るよ

うになつており、テープもついている。　テープには、発声されたロシア語が入つており、コンピュータが学生に対して、英語からロシア語、あるいはロシア語から英語へと書く形と話す形で練習をさせるわけである。

このロシア語を教えるプログラムは非常に精巧であつて、ロシア語のコース全体の3分の2をコンピュータで教え、3分の1は先生が直接教える形をとつている。　普通に教えた場合には、4分の3位が脱落していなくなり、試験に合格できないが、コンピュータで教えた場合は4分の1しか脱落しなかつた。　したがつて、今年度は、ロシア語を全面的にコンピュータで教えるほうに切りかえることになり、また、2年目のロシア語もコンピュータで教えようという動きがある。　上にあげた例では、種々のことができるように考えられるが、コンピュータで教えるためには、現在の段階ではまだいろいろな制限がある。理由は2つ考えられるが、1つの理由は、現在あるような非常に簡単な教育用のプログラムでは、コンピュータで教えられるような科目が限られていること。　2つ目の問題点は費用の点である。　経済性という点では、コンピュータで教えるための費用は莫大な金額となる。

しかし、私の考えでは、特に教育用に設計したコンピュータならば、現在の技術でも経済的なものができるはずである。　現在は用途が非常に限られているためその点を打開するプログラムを多く作らなければならない。

次にアプリケイションについて述べると、タイムシェアリング用のアプリケイションでなくてコンピュータを制御用に使用する方法がある。　自動車の制御に関しては、いろいろな観点があるが1つの考え方は、大型コンピュータを用いて、ある地方あるいは地域に走つている車を、全部コンピュータで制御することである。　もう1つのは小型コンピュータを、各々の車に積み自動車の運転手の代わりに使用するものである。　この自動車の制御に関して、詳しく述べることはここではできないが、私達は1つのコンピュータで1台の自動車を制御する実験を行なつている。　この方法は、現在では非常に不経済であるがコンピュータが、安くなりプログラム作成料が安くなると非常に有効となろう。

基本的な問題は、自動車の窓から見える景色を視覚的にとらえることである。　具体的には道路の上のものをテレビカメラのようなもので見て道路の端とか、道路の交差点、障害物、その他の車、歩行者等を見分けて位置を判断し道路標識を読むことである。

上に述べた問題は困難ではあるが解決不可能ではない。　コンピュータによる車の制御の利点は幾つかあるが、その1つはシステムを用いて非常に安全な運転ができることである。　人間よりも安全な運転をして、人間がやつている場合よりも事故を減らし、現在の

アメリカ国内における事故率を１０分の１以下に減らそうと考えている。

もう１つの利点は、高速道路や普通の道路をもつと有効に使用できることである。道路が、どの程度混雑するかは上で述べた人間の応答時間に大いに関係があるわけで、コンピュータによるシステムでは時速１２０kmで車がギッシリ詰まつていても現在の常識よりも安全に運転が出来そうに思われる。

３番目の利点は、車を自由に何処へでも動かせることである。　たとえば車の保守に行かせたり仕事のする場所に行き、車を駐車場へ行かせることもできる。　帰りにはまた車を駐車場から仕事場の戸口に迎えにこさせ、丁度おかかえの運転手を雇つているように、使用することができる。　また、運転ができない人にとつても車が使えるということもあり子供とか老人といつた人達が車を使えることにもなる。

もう１つ、私が最後に申し上げたいトピックは、各家庭にコンピュータのコンソールを持ち込むことであるが、現在実際自宅に持つている人もおり、プログラムを書いたことのある人もいる。　しかし、これには障害になる問題点がある。

その１つは、タイムシェアリングがまだ充分ではないということ。　また、コンソールが非常に高いということ。　さらに、各家庭に持ち込んで使用する場合を想定するとピッタリしたアプリケイションがないことが上げられる。　最初の２つの問題点は、時と共にだんだん改善されていくと考えられるが３番目の問題については、まだまだこれからの問題である。

１つの使い方としては、コンピュータを各家庭に持ち込んで、ものを教えることに使用する方法がある。　たとえば、ある特定の飛行機の切符があるかないかということを、コンピュータやコンソールから問い合わせる。　また、ある特定のものが、現在店にあるかどうかということを調べるのに使用するといつた方法もある。　こういつた最後の２つのような使い方は、現在やつと集団的な規模で使われ始めているきざしがある。たとえば、１日１００ドル出すとコンソールをつけられるようになり、広く用いられるようになるのではないかと考えられるが、まだ将来のことがあろう。

これで私のタイムシェアリングについての話を終わることにする。

請求番号 NPOEC 43-9	登録番号
著者名 タイムシェアリングの原理と応用 スタンフォード大学教授	
書名 工学博士 JOHN・McCARTHY	

所属	帯出者氏名	貸出日	返却予定日	返却日