超並列計算機を用いた実例型翻訳の実現

An Example-Based Translation System on a Massively Parallel Machine

佐藤理史

Satoshi SATO

北陸先端科学技術大学院大学

Japan Advanced Institute of Science and Technology, Hokuriku

sato@jaist-east.ac.jp

Abstract

This paper describes MBT3n, MIMD implementation of MBT3 on nCUBE2. MBT3 is an Example-Based Translation system for technical terms. In MBT3n, *parallel best match retrieval* is used while *sequential best match retrieval* is used in the original (sequential) MBT3 system. The translation performance of MBT3n improves as the number of processors MBT3n runs on increases. MBT3n with 256 processors on nCUBE2 is about ten times faster than the original MBT3 system on SparcStation2 in the best.

1 はじめに

実例型翻訳 (Example-Based Translation) とは、翻訳例を模倣利用することによって翻訳を実現しようとする、新しい翻訳方式の一つである [佐藤, 1992]。1981 年の長尾による最初の提案 [Nagao, 1984]を受け、1988 年頃より研究が盛んに行なわれてきている。

MBT3 [Sato, 1993a] は、MBT1 [佐藤, 1991a]、MBT2 [佐藤, 1991b] に続く、筆者の実例型翻訳の最新システムである。本システムは、翻訳における困難な問題の 1 つである、専門用語の翻訳を行なうために設計された。予備的な実験において、MBT3 は、データベース中に格納した既知の用語に対しては 99%、データベース中に存在しない未知の用語に対しては 78%の翻訳正解率を示した。

MBT3 における研究課題の 1 つは、実例型翻訳の並列化を検討し、以下の疑問に答えることにある。

Q1. どのような並列計算モデル、あるいは、アーキテクチャが、実例型翻訳に向いているか？

Q2. 実例型翻訳のどの部分を並列化すべき／できるか？

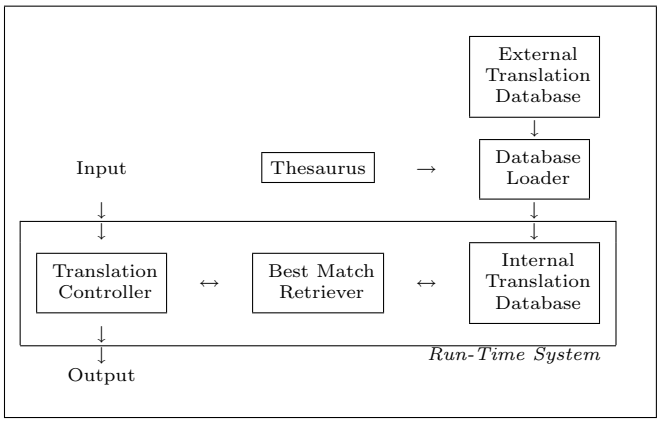

Figure 1: The Organization of MBT3

Q3. 並列版実例型翻訳は、逐次版実例型翻訳に比べて、どの程度速くなるか？

Q4. 翻訳速度は、プロセッサ数に比例して向上するか？

Q5. 並列版実例型翻訳では、どのくらいのサイズのデータベースを使用可能か？

本論文では、MIMD 計算機である nCUBE2 上の MBT3 の実現法—MBT3n—を提案し、これらの疑問に答えることを試みる。

2 MBT3 の概要

MBT3 の概要を Figure 1 に示す。MBT3 の実行系は、1) 内部翻訳データベース、2) 最適照合検索器、3) 翻訳コントローラの 3 つのモジュールからなる。

MBT3 による翻訳処理は、おおよそ、以下のように進む。

1. 翻訳入力が与えられる。
2. それを検索入力とし、最適照合検索を実行する。その結果として、データベースから検索入力に良く似た $N(= 10)$ レコードを得る。
3. 得られた N レコードから、最良の翻訳パタンを選択する。

```
MBT>下降型構文解析プログラム
*** Input ***
                    [ 下降 型 構文 解析 プログラム ]
Retrieve...
*** Target Pattern Ranking ***
1:(1448): [ [ 下降 型 ] [ 構文 解析 プログラム ] ]
2:(1371): [ [ top-down ] [ 構文 解析 プログラム ] ]
3:( 749): [ [ 下降 ] [ 型 ] [ 構文 解析 プログラム ] ]
*** Input ***
                    [ 下降 型 ] 構文 解析 プログラム
Retrieve...
*** Target Pattern Ranking ***
1:( 561): [ [ top-down ] ]
2:( 252): [ [ 下降 ] [ type ] ]
3:( 158): [ [ 下降 ] [ 型 ] ]
*** Input ***
              下降 型 [ 構文 解析 プログラム ]
Retrieve...
*** Target Pattern Ranking ***
1:( 981): [ [ parsing ] [ プログラム ] ]
2:( 900): [ [ syntactic analysis ] [ program ] ]
*** Input ***
    下降 型 構文 解析 [ プログラム ]
Retrieve...
*** Target Pattern Ranking ***
1:( 276): [ [ program ] ]
2:( 117): [ [ programs ] ]

*** Final Result ***
                    [ 下降 型 構文 解析 プログラム ]
                    [ top-down parsing program ]
```

Figure 2: Translation Process

4. 翻訳パタンが完全な翻訳となっていない場合は、まだ翻訳されていない部分を翻訳する。(この手続きを再帰的に呼び出す。)
5. 最終的な翻訳出力を得る。

Figure 2 に、翻訳処理のトレースを示す。詳細は、文献 [Sato, 1993a] を参照してほしい。

3 MBT3n: nCUBE2 上の MBT3

MBT3n では、最適照合検索を実行する部分 (前節ステップ 2) を並列化した。

3.1 プロセッサの割り当て

nCUBE2 は、ハイパーキューブ結合の MIMD マシンである。プログラムは、2^d 個のプロセッサから構成されるサブキューブで実行することができる。それぞれのプロセッサには、0 から $2^d - 1$ までのプロセッサ ID が付けられている。ここでは、プロセッサ 0 をホストプロセッサと呼び、他のプロセッサをノードプロセッサと呼ぶ。ホストプロセッサは、翻訳コントローラとして動作するのに対し、ノードプロセッサは並列最適照合検索でのみ用いられる。

3.2 並列最適照合検索

並列最適照合検索の大枠は、以下で与えられる。

1. データベースをいくつかのサブデータベースに分割し、それぞれを、ノードプロセッサのローカルメモリに格納しておく。

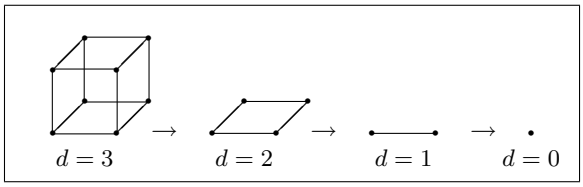

Figure 3: Reduction of a Hypercube dimension

2. 各プロセッサ上で、最も似ているレコードを N レコード見つける。
3. 全てのプロセッサの検索結果から、全体の検索結果を得る。

この大枠において、ステップ 3 の実現法が問題となる。ある 1 つのプロセッサで最終検索結果を求める処理を行なおうとすると、そのプロセッサにメッセージと負荷が集中し、スピードアップが図れない。

この問題に対する我々の解は、以下のような並列縮退法である。その基本的なアイディアは、ハイパーキューブの次元の縮退から来ている (Figure 3)。例えば、3 次元のハイパーキューブを考えよう。そのそれぞれのプロセッサの ID を 3 桁の 2 進数 $b_2b_1b_0$ と表すことにしよう。まず、最初に、奇数の ID を持つプロセッサは、偶数の ID を持つプロセッサに、検索結果を送信する。

$$\begin{array}{cccc} 001 & 011 & 101 & 111 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 000 & 010 & 100 & 110 \end{array}$$

偶数の ID を持つプロセッサは、受けとった検索結果と自分自身の検索結果から、新しい検索結果を計算する。今、各プロセッサが N レコード検索すると仮定すれば、ここでは、$2N$ から上位 N レコードを選ぶことになる。この処理によって、$8N$ レコードが $4N$ レコードに縮退する。これは、b_0 を縮退したことに相当する。

次に、b_1 を以下のように縮退する。

$$\begin{array}{cc} 010 & 110 \\ \downarrow & \downarrow \\ 000 & 100 \end{array}$$

最後に、b_2 を縮退する。

$$\begin{array}{c} 100 \\ \downarrow \\ 000 \end{array}$$

この結果、最終的な検索結果がプロセッサ 000、つまり、ホストプロセッサに得られることになる。なお、この処理は、非同期に実行することができる。例えば、プロセッサ 100 は、プロセッサ 101 と 110 からそれぞれメッセージを受けとるが、その順序は、どちらが先でも良い。重要なのは、メッセージの数だけである。

以下に、完全な並列最適照合検索の手続きを示す。

I. ホストプロセッサ上
   1. 検索入力を全てのノードプロセッサにブロードキャストする。
   2. 自分自身の検索結果を空にする。
   3. d 回繰り返す (ここで、d は、ハイパーキューブ次元である)。
      (a) 検索結果を他のプロセッサから受けとる。
      (b) それを自分自身の結果とマージし、その中から、上位 N 個を選んで、新しい自分自身の検索結果を作る。
   4. 自分自身の検索結果が、求める最終的な検索結果となる。

II. ノードプロセッサ $b_{d-1}\cdots b_k0\cdots0$ 上 (b_k は、非 0 となる最下位ビット)
   1. 検索入力を受けとる。
   2. 自分が記憶しているサブデータベースから、検索入力と最も良く似た N レコードを求める。それを、自分自身の検索結果とする。
   3. k 回繰り返す。
      (a) 検索結果を他のプロセッサから受けとる。
      (b) それを自分自身の結果とマージし、その中から、上位 N 個を選んで、新しい自分自身の検索結果を作る。
   4. 自分自身の検索結果をプロセッサ $b_{d-1}\cdots b_{k+1}0\cdots0$ に送る。

4 実験

MBT3n と MBT3s[1] の性能比較の実験を行なった。この 2 つのシステムは、全く同じ翻訳結果を出力するので、その違いは、翻訳速度のみである。

実験では、2 つのデータベースを用意した。一方は、11317 レコードからなり、他方は、19669 レコードからなる。後者は、前者を完全に含んでいる。翻訳の対象とする入力セットも 2 つ用意した。一方の *info* は、94 個の専門用語からなり、他方の *prog* は、107 個の専門用語からなる。この 2 つのセットには、重複はない。

これらのデータベースと入力セットを用いて、翻訳に必要な時間を測定した。その結果を Table 1 に示す。MBT3n は、可能な全てのプロセッサ数に対して、翻訳時間を測定した。なお、ノードプロセッサの数は、(プロセッサ数 -1) であることに注意してほしい。

これらの結果から、以下のことが観察できる。

1. 並列最適照合検索は、高速化に非常に効果がある。256 プロセッサ使用時の MBT3n は、MBT3s に比べて、最高で約 10 倍高速である。

[1] 逐次版 MBT3 システム。SparcStation2 で動作する。

Figure 4: Number of Node Processors vs. Translation Time

2. 8 プロセッサ使用時の MBT3n は、MBT3s とほぼ同等の性能である。
3. プロセッサの数を増やすと、翻訳時間は減少する。Figure 4 に、ノードプロセッサ数と翻訳時間の関係のグラフを示す。プロセッサ数が 127 まではほぼ直線的に翻訳時間が減少する。
4. この現象は、プロセッサ当たりのレコード数と翻訳時間の関係のグラフ (Figure 5) でも確認できる。プロセッサ当たりのレコード数を 300 まで減らしていくと、翻訳時間もそれに比例して減少する。それ以上減らすと、その効果は、少しずつ減退していく。
5. データベースを 1.74 倍にしても、逐次版の MBT3s でも翻訳時間は、1.74 倍にはならず、それよりは短くなる。これは、最適照合検索の実行回数の減少に起因している。一般に、データベースをサイズを大きくしていくと、より大きな単位での翻訳が増えるため、最適照合検索の実行回数が減少する。Table 1 の BMR の欄は、最適照合検索の実行回数を示している。
6. 比率 b/a は、プロセッサ数の増加に従って減少する。ここで、a は、小さなデータベースを使った場合の翻訳時間であり、b は、大きなデータベースを使った場合の翻訳時間である。特に、プロセッサ数が 64 以上では、この比率がかなり小さくなる。

並列最適照合検索に必要な計算時間は、各ノードプロセッサで実行される逐次最適照合検索の時間の最大値に依存する。逐次最適照合検索に必要な計算時間は、そのサブデータベース中で、照合の得点を計算しなければならないレコードの数に依存する[2] 。照合の得点を計算しなければならないレコードが各サブデータベースに平均的に

[2] 逐次最適照合検索において、各レコードに対する処理は、1) 照合するかを調べ、2) 照合する場合にその照合の得点を求める、という 2 ステップで行なわれる。前者は非常に短時間で調べることができるの対して、後者はかなり時間がかかる。

Table 1: Result of Experiments

Input Set	DB size	BMR	MBT3s	MBT3n 4	8	16	32	64	128	256	
info	11317	200	90	333	92	74	29	25	15	14	a
	19669	180	138	476	149	103	42	30	18	15	b
	1.74	0.90	1.44	1.42	1.62	1.39	1.45	1.20	1.20	1.07	b/a
prog	11317	225	108	407	111	90	35	30	18	16	a
	19669	197	177	564	175	121	49	37	21	18	b
	1.74	0.88	1.64	1.39	1.58	1.34	1.40	1.23	1.17	1.13	b/a

Figure 5: Number of Records per Node Processor vs. Translation Time

分散している時、MBT3n は最大性能を示す。各サブデータベースのレコード数が十分に大きい時は、このような分散が期待できるが、レコード数が少なくなるにつれて、偏りが生じてくる。これが原因となって、スポードアップが減退する。

5 議論

以上の実験結果を踏まえ、冒頭に示した 5 つの疑問に答えよう。

Q1. どのような並列計算モデル、あるいは、アーキテクチャが、実例型翻訳に向いているか？
A1. MIMD は実例型翻訳に向いている。

今回の実験結果は、最初の MIMD インプリメンテーションとしては、非常にうまくいったと考えられる。これに先立ち、CM-5 上の C*を用いて、MBT3 の SIMD インプリメンテーションを試みたが、その結果は、我々を非常に失望させるものであった。現時点では、SIMD が実例型翻訳に向かないという結論をまだ下してはいないが、かなり悲観的な感触を持っている。

Q2. 実例型翻訳のどの部分を並列化すべき／できるか？
A2. 最適照合検索を並列化すべきであり、かつ、それは可能である。

最適照合検索に要する計算時間は、実例型翻訳システムの実行時間のほとんどを占める。故に、この部分を並列化することによって、十分な高速化をはかることができる。

Q3. 並列版実例型翻訳は、逐次版実例型翻訳に比べて、どの程度速くなるか？
A3. 256 プロセッサ使用時の MBT3n は、MBT3s に比較して、最高で約 10 倍高速であった。

データベースがより大きくなった場合、さらなる高速化が期待できる。

Q4. 翻訳速度は、プロセッサ数に比例して向上するか？
A4. プロセッサ当たりのレコード数が十分多い場合は、プロセッサ数に比例して翻訳速度は向上する。

MBT3n においては、プロセッサ当たりのレコード数が 300 以上であるとき、翻訳速度は、プロセッサ数に比例して向上する。

Q5. 並列版実例型翻訳では、どのくらいのデータベースを使用可能であるか？
A5. 現在の見積りによれば、数百万レコードのデータベースが使用可能である。

見積りは、以下の通り。nCUBE2 の最大プロセッサ数は、8192 である。各プロセッサに 256 レコードのデータを格納するとすると、システム全体では、$8192 \times 256 = 2097152$ のデータを格納することができる。

これらの答えは、すべて、並列版実例型翻訳が有望であることを示している。

参考文献

[Nagao, 1984] Nagao, M.: A Framework of a Mechanical Translation between Japanese and English by Analogy Principle, in Elithorn, A. and Barnerji, R. (Eds.), *Artificial and Human Intelligence*, North-Holland, pp.173–180, 1984. (*Proc. of the International NATO Symposium on Artificial & Human Intelligence*, 1981, Lyon, France.)

[佐藤, 1991a] 佐藤理史: MBT1: 実例に基づく訳語選択, 人工知能学会誌, Vol.6, No.4, p592–600, 1991.

[佐藤, 1991b] 佐藤理史: MBT2: 実例に基づく翻訳における複数翻訳例の組合せ利用, 人工知能学会誌, Vol.6, No.6, pp861–871, 1991.

[佐藤, 1992] 佐藤理史: 実例に基づく翻訳, 情報処理, Vol. 33, No.6, pp673–681, 1992.

[Sato, 1993a] Sato, S.: *Example-Based Translation of Technical Terms*, JAIST Research Report, IS-RR-93-4I, School of Information Science, Japan Advanced Institute of Science and Technology, Hokuriku, 1993. (Revised version is in *Proc. of the fifth International Conference on Theoretical and Methodological Issues in Machine Translation*, Kyoto, to appear.)

[Sato, 1993b] Sato, S.: *MIMD Implementation of MBT3*, JAIST Research Report, IS-RR-93-5I, School of Information Science, Japan Advanced Institute of Science and Technology, Hokuriku, 1993. (本稿は、この文献の日本語短縮版である。)