

繁説エージェントによる分散システムフレームワークと 情報検索システムの構築

山本英雄 梅村恭司
豊橋技術科学大学 情報工学系

分散処理するとき、処理単位は、それぞれ単数の依頼、実行、回答であることが多い。回答を複数行なり記述をすることも可能であるが、それは例外的な状況と考えられている。しかし、これを例外と考えることも可能である。本稿では処理に対する回答を一度にせず、ある条件が満たされた都度に行なり分散処理のフレームワークを提案する。そして、それが有効な条件をモデル化とシミュレーションで確認し、提案するフレームワークにしたがった情報検索システムを構築した結果を報告する。

A Framework for Verbose Agent and its Experience of Information Retrieval System

Hideo Yamamoto and Kyoji Umemura
Department of Information and computer science
Toyohashi University of Technology

The fundamental job for distributed processing is the set of request, process, and reply. Though multiple reply is possible, it is generally considered as a rare case. This article proposes a framework for a distributed system where replies are issued whenever certain condition is satisfied. Our simulation shows when the multiple reply is beneficial, and our application system of information retrieval verifies the feasibility.

1 はじめに

近年、分散された大量のデータを対象とした検索が行なわれるようになった。また、そのためのエージェントのアイデアが注目されている [1][2]。また、我々は、そのためのシステムの実装方法を研究してきた [8][9]。分散情報検索では、情報のすべてを調べ尽くすことが実際には難しくなっている。そして、途中結果を速やかに入手することが利益がある。我々はそのような検索を、検索範囲が無限大、検索結果も複数、途中結果も有効であるという性質を持つ問題とモデル化し、そのような条件を満たす検索には、途中結果の報告と検索の打ち切り制御が有効であると考えた。これは人間ならば実際に行なっている検索法だが、コンピュータシステムで行なわれることが少ない方法である [3][4][5] [6][7]。

新聞記事検索は、そのリソースの大きさから候補の記事が複数存在することから上記の検索問題に合致する。そこで我々は、オンラインの新聞記事検索システムを構築するために、途中結果の報告の機能を備えた分散システムフレームワークを作成し、それを用いて新聞記事検索システムを構築し、そのフレームワークの妥当性を確かめた。

2 対象とする検索問題の性質と方針

ここではまず、我々が対象とする問題の条件をまとめる。対象とする問題の性質として、以下に挙げるものを示した。「(1) 検索範囲は実効的に無限大、(2) 検索結果は複数、(3) 途中結果も有効」上記の条件の妥当性を新聞記事検索を例として説明する。

検索範囲は無限大: 1つの新聞記事リソースは有限であるが、新聞記事リソースは、年度や新聞社などによっていくらかでも追加することができる。つまり、ある新聞記事リソースを検索し終ったとしても、次の新聞記事リソースを探しに行くとしたら、実効的にいつまでも探し続けることができる。

検索結果は複数: 新聞記事検索では、ユーザが欲しい記事に対していくつかの候補記事が得られる。例えば企業合併という入力に対して、銀行の合併記事や鉄鋼会社の候補記事が得られる。

途中結果も有効: あいまいな入力を許すようなコンセプトの検索では、その性質上唯一の解というものとは存在しない。つまり、入力に対して99%の類似度で一致する記事があったとしても、その検索過程で見つかった80%の類似度で一致する記事もユーザが求める記事である可能性がある。

3 性能評価のためのシミュレーション

条件を満たす問題に対しては、途中結果の報告と打ち切り時間を導入して、速やかにサンプルを得ることや、その区間で何らかの情報を得られることは有効だという直観は働く。しかし、実際にはトレードオフが存在する。もし報告の処理によるコストが生じないのであれば、途中結果を報告した方が有利であることは明らかである。また、もし検索のために無限に待つことができるという仮定があるなら、途中結果を報告しようがしまいが、最終的な解の結果は同じになる。しかし実際には、報告の処理によるコストが存在し、検索のために無限に待てるというのは現実的でない。そこでまず、我々のアプローチが有効である条件をシミュレーションによって確かめる。

3.1 解発見モデル

今回のシミュレーションでは、検索過程における次の解の候補が得られるまでの時間間隔をモデル化した。以後これを解発見モデルと呼ぶことにする。解発見モデルには、おおよそ等間隔で解を発見できるが、ある程度の分散を持つモデルを選んだ。今回のシミュレーションでは、次の解の候補が得られるまでの時間間隔は次式で表される確率密度関数で表すこととした。

$$f(x) = xe^{-x}$$

$f(x)$: 解発見の時間間隔の確率密度関数
 x : 時間間隔 ($x > 0$)

今回のシミュレーションでは、解発見モデルにバリエーションを持たせるために、次式のように変数変換を施したモデルを使用した。図1にいくつかの解発見モデルを示す。

$$y = ax, f(y) = f(ax) = \frac{1}{a^2} xe^{-\frac{y}{a}}$$

$f(x)$: 解発見の時間間隔の確率密度関数
 x : 時間間隔 ($x > 0$)
 a : モデルパラメータ

図1に示されるとおり、次の解が見つかるまでの間隔の平均が大きいモデル程、解を得るまでの期待値が大きいことが分かる。

3.2 解のスコアのモデル

解発見モデルによって得られた時間間隔で得られる解のスコアは、0から100の間の値をとり、一様分布に従うものとした。

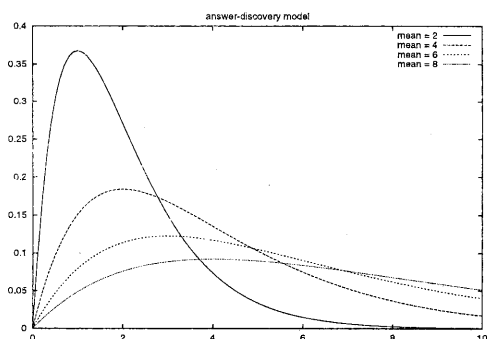


図 1: 解発見モデル

3.3 検索モデル

単報告検索と複報告検索のシーケンスは図 2 のようになる。単報告検索では、ユーザへ解を報告するために生じる処理は、解を発見したときのみである。対して複報告検索の場合、ある間隔ごとに報告するように設定してあるとすると、その間隔ごとに報告による処理が生じる。

もし報告による処理を無視することができるならば、単報告検索でも複報告検索でも、解を発見するまでに要する時間は同じとなり、複報告検索が有利となる。しかし実際には、各報告ごとに余分な処理が加わり、複報告検索の方が解を発見するまでに多く時間を要する。そこで、評価においては、時間に対する解の質を考え、遅延をパラメータとしてシミュレーションを行なう。

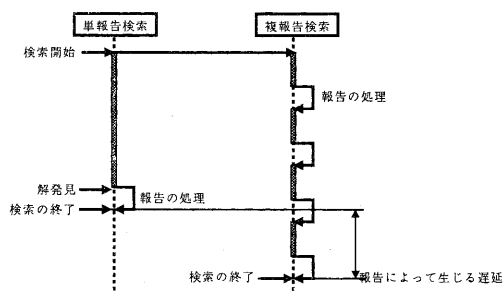


図 2: 遅延がある場合

3.4 遅延をパラメータとした場合の性能評価

遅延をパラメータとした場合の性能評価では、単報告検索で解が発見される以前に、複報告検索の途中結果の報告によって何らかの情報が得られる確率が 50 % となるような、解発見の平均間隔と遅延の大きさの比率をシミュレーションによって求めた。表 1 に結果を示す。

表 1: 遅延による性能評価結果

解発見の間隔	分散	遅延時間	相対時間
2	2	11.362	5.681
4	8	22.204	5.551
6	18	33.672	5.612
8	32	48.264	6.033

表 1 から、遅延の大きさが解発見の平均間隔よりも 5 ~ 6 倍のときが、報告の前に情報を得ることのできる確率と得ることのできない確率の分岐点であることが示されている。

3.5 打ち切り時間をパラメータとした場合の性能評価

遅延をパラメータとした場合のシミュレーションでは、報告された解のスコアは考慮しておらず、単報告検索において解が得られるまで無限に待つことができるという仮定でシミュレーションを行なった。そこで、検索に打ち切り時間を導入し、その期限内の検索で得られる解の期待値をシミュレーションし、比較することにする。

このシミュレーションでは、打ち切り時間をパラメータとし、両検索で打ち切り時間内に得られるスコアの期待値を調べる。このとき、単報告検索では 99 % 以上のスコアを持つ解を条件を満たす解とし、複報告検索では、打ち切り時間内で見つかった解候補のスコアの最大値を期限内で得られた解としている。なお、遅延の大きさは前回のシミュレーションの結果を用いる。

図 3 に、解発見平均間隔 8,000[msec]、遅延時間 48,264[msec] である場合のシミュレーション結果を示す。グラフの横軸は打ち切り時間 [msec]、縦軸は打ち切り時間内に得られる解の期待値である。

図 3 より分かることは、もし打ち切り時間が遅延時間よりも短かった場合、複報告検索では、解が見つかったとしても報告できない分不利である。

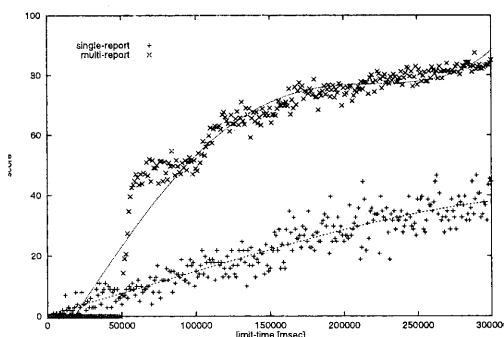


図 3: 打ち切り時間による性能評価結果

しかし打ち切り時間がそれよりも長かった場合、遅延時間が解発見平均間隔の5～6倍であったとしても、一回でも報告できれば、複報告検索の方が高い期待値を得ていることが分かる。実際には、解を通知する遅延時間と解の発見の間隔比較では解発見に時間がかかるものを想定していれば、途中結果の報告が有効であるというシミュレーション結果が得られた。

4 分散システムフレームワーク

本フレームワークには以下の4つの基本概念が存在する。報告条件が我々のフレームワークの特徴となっている。

ワーカー (Worker): 本フレームワークでは、ワーカーの集まりに対して処理を分散する。ワーカーとはエージェントの到着地であり、エージェントが運んで来るジョブを処理する役割を担当している。ワーカーはエージェントにジョブの実行をサービスとして提供しているともいえる。ワーカーには大きく4つの役割がある。(1) エージェントの受け入れ、(2) エージェントからのジョブの受取りと実行、(3) エージェントの移動を記録、(4) ジョブの結果報告の処理、以上の4つである。

エージェント (Agent): 本フレームワークでは、分散させたい処理をエージェントによって分散させる。エージェントの役割は、単にユーザから受け取ったジョブを持ってワーカー間を移動することである。ただしエージェント自身には移動のための機能はなく、移動もジョブとしてエージェントが保持する。それぞれのエージェントはワーカーに駐在することができ、ワーカーに自分の保持するジョブを処理してもらえる。

ジョブ (Job): 本フレームワークでは、エージェントによって運ばれる処理をジョブと呼ぶ。ジョブはエージェントによって運ばれ、ワーカーによって処理される。

報告条件 (ReportCondition): 本フレームワークは、エージェントに課されたジョブが全て終了していなくとも、途中結果を報告できるように設計されているのが特徴である。その報告の仕組みに用いられるものが報告条件である。この報告条件という概念が一般的な分散システムフレームワークにはない概念である。報告条件には、どういった時に、どういった条件で途中結果を報告するのかを記述する。報告を行なう間隔や、ある条件を満たした結果のみを報告させるための条件などを設定することができる。この報告条件は、エージェントとジョブ両方、またはいずれかに設定することができる。デフォルトの実装では、ジョブに設定された報告条件が優先されるので、エージェントが保持するジョブに共通した報告条件はエージェントに設定して、あるジョブのみ報告条件を変えたいときには、そのジョブに異なった報告条件を設定するような使い方ができる。

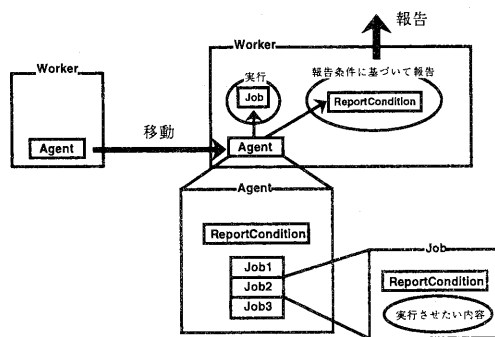


図 4: フレームワーク解説

図4に処理の流れを示す。まず分散させたい処理をジョブとして実装する。次に途中報告をする条件を報告条件として実装する。そしてジョブと報告条件をエージェントにセットし、エージェントワーカーへ送り出す。各ワーカーへ移動したエージェントは、ワーカーにジョブを実行してもらい、報告条件に従って、ジョブの途中結果を報告する。

5 コーディングの具体例

ここでは以下のような状況を想定する。host1、host2、host3へ次々に移動して、それぞれのホス

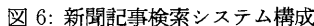
(1)host1 では 30 秒ごとに報告するが、報告対象となるのは類似度 0.03 以上のものだけとする。(2)host1 以外のホストでは 60 秒ごとに報告するが、報告対象となるのは類似度 0.001 以上のものだけとする。図 5 にこれらのクラスを使ったコーディング例を示す。

1-2行目で報告条件オブジェクトを生成している。それぞれ、報告頻度 30 秒、最低類似度 0.03 の報告条件と、報告頻度 60 秒、最低類似度 0.001 の報告条件を表している。3行目ではエージェントを生成している。ここでは、host1でのジョブ以外に適用される報告条件(報告頻度 60 秒、最低類似度 0.001)を同時に設定している。4行目ではhost1へ移動するためのジョブをエージェントに設定し、5行目でhost1で行なう検索のジョブを設定している。ここでは、host1でのみ有効な報告条件を設定するために、ジョブに報告条件を設定している。以下同様に、host2へ移動、検索、host3へ移動、検索という順番でジョブをエージェントに設定している。

提案するフレームワークの妥当性を検証するために新聞記事検索システムを作成した。新聞記事検索は、巨大な記事アーカイブが対象であり、結果の候補記事も複数、検索仮定の候補記事も有効であることから、我々の対象とする検索問題に合致する。このシステムでは、あいまいな入力 of 記述を許し、検索対象となるリソースについても自然言語で書いたものを検索するようなアルゴリズムを使用している。しかしながら、柔軟な入力を許す反面、その計算量は多く、過去数年分の新聞記事を検索するような場合には多くの時間を要する。

新聞記事検索システムは図6のように構築される。本システムでは、分割・分散された新聞記事アーカイブ各々がエージェントの移動先となる。

図 5: サンプルコード



エージェントは各新聞記事アーカイブへ移動し、そこで新聞記事検索プログラムを実行する。実行の途中結果がユーザに報告されるが、報告の条件は、設定された閾値以上の類似度を持つ候補記事が見つかったときと、以前報告した候補記事より類似度の高い記事が見つかったときの2つである。

6.2 実行環境

新聞記事検索システムの実行環境は、表2に示される7台のマシンより構成される10Base-Tで接続されたネットワークである。新聞記事アーカイブは、毎日新聞94-95年度版(約200MB)を7つに分割・分散させている。

表 2: 実行環境

PentiumII 400MHz Memory 512MB	× 1
PentiumII 300MHz Memory 256MB	× 1
PentiumII 300MHz Memory 32MB	× 3
AMD K6 333MHz Memory 256MB	× 2

6.3 実行時間

ある検索実行時間を表3に示す。入力に対しては、52個の候補記事が得られる。途中報告をする場合でもしない場合でも、52個すべての候補記事を得るための要する時間に大きな差はない。これは、報告の頻度を減らすために30秒ごと結果を報告するように設定してしており、報告の回数も軽減している。かつ、この30秒は途中結果を報告するのに必要な時間よりも十分に長い。さらに途中報告をする場合の実行では、検索開始から30秒経過した時点で3つの候補記事が得られた。これは報告条件が30秒ごとに結果を報告するように設定してある場合の1回目の報告にあたる。

表 3: 実行時間

実行環境	実行時間 [sec]
K6 333MHz	2196
途中報告なし	471
途中報告あり	477
(全ての候補記事)	
(最初の候補)	30(一回目)

7 まとめ

我々は、検索範囲は無限大、検索によって得られる解は複数、検索の途中結果も有効の3つの条件を満たすような検索問題に対して、途中結果の報告と検索の打ちきりのアプローチを提案した。そして、シミュレーションによって、提案する方法が従来の方法による検索よりも有効である条件を確かめた。

また、このアプローチに基づいた分散処理のためのエージェントフレームワークを作成し、実際に応用システムとして新聞記事検索システムを構築した。分散した記事リソースを対象とすることができ、ユーザへ見つかった記事候補を逐次報告できるシステムである。その結果、応答を早めに見ることができることなど、今回作成したフレームワークが有効となることを確認できた。

参考文献

- [1] 本位田真一, 大須賀昭彦: オブジェクト指向からエージェント指向へ, ソフトバンク, 1998.
- [2] Eun-Seok LEE, Roberto Okada, Norio Shiratori: Agent-based Social Information Gathering on Internet, ICMAS-96, p448, 1996.
- [3] 山崎重一郎, 津田宏: Telescript 言語入門, アスキー出版, 1996.
- [4] Bal, H. E., Steiner, J.G., and Tanenbaum, A.S.: Programming Languages for distributed computing systems, ACM Computing Surveys 21(3), 1989
- [5] Birrell, A.D. and Nelson, B.J.: Implementing remote procedure calls, ACM Transactions on Computer Systems 2(1), pp 39-59, 1984.
- [6] 小野沢 博文: 分散オブジェクト指向技術 CORBA, ソフト・リサーチ・センター, 1996.
- [7] Jamusz Kowalik: PVM: Parallel Virtual Machine A User's Guide and Tutorial for Networked Parallel Computing, MIT Press, 1994.
- [8] 下川僚子, 山本英雄, 梅村恭司: トランスレータ方式のエージェント言語の実装情報処理学会研究報告 98-OS-78, pp31-38, 1998
- [9] 下川僚子, 梅村恭司: トランスレータを利用した機種非依存な実行移送方式情報処理学会論文誌, 1999年6月, 掲載予定