システムソフトウェアと
オペレーティング・システム 84−20
（2000．5．25）

繁説エージェントからの途中報告の処理

森田卓也† 山本英雄‡ 梅村恭司†
†豊橋技術科学大学 情報工学系 梅村研究室
‡NTT

エージェントを用いた分散処理は，エージェントに対し仕事を依頼し，仕事がすべて完了してから結果報告をエージェントから受けとる．しかし，分散処理の処理時間が長い場合など，処理途中の状態を知りたい場合があることが多いと我々は考えている．そこで我々は，ある条件が満たされる都度に途中報告を行う”繁説エージェント”を考案し，途中報告を行う分散システムフレームワークを構築した．このフレームワークにおいて，途中報告が頻繁に行われると，受け取り側の処理は単数の報告に比べ複雑になってしまう．本稿では，複数回行われる途中報告を受け取る処理について提案する．そして実際に新聞検索システムに応用し，途中報告を行わせユーザへ示す意義を説明する．

Dealing with Report from Verbose Agent

Takuya Morita†, Hideo Yamamoto‡ and Kyoji Umemura†
†Department of Information and Computer Sciences
Toyohashi University of Technology
‡NTT

Distributed processing systems usually receive results from agents after the agents finish their tasks. However, we frequently need to know the status in the middle of the process. We have proposed “Verbose Agent” who replies in the middle of the process, as a framework for distributed system where replies are issued whenever certain condition is satisfied. In this framework, the receiver becomes complicated when there are lots of replys from many agents. This paper addresses this problem of Verbose Agents and its solutions. We have built a newspaper-article retrieval system using our framework, and show usefulness of our framework by building an application.

1 はじめに

近年，エージェントとを用いて分散処理を行なわせる研究が頻繁に行なわれている．ここで言うエージェントとは，他の計算機に対し移動して走行するプログラムのことである．分散処理を行なうときの処理単位は，それぞれ単数の依頼，実行，回答である場合が多い．しかし我々は，実行の処理に時間がかかる場合などには，途中までの部分回答も重要な意味を持つと考えた．

そこで，我々は処理に対する回答を一度にせず，ある条件が満たされる都度に途中報告を行なう分散処理のフレームワークを提案した (文献[2])．更に，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，フレームワークの重要性を確認した．

様々な仕事の途中報告が頻繁に行なわれることにより，その途中報告の受け取り側の制御が複雑になってしまうことが実システムを作成するときに問題となった．本稿では，報告の受け取り側に着眼点を置き，どのように途中報告を受け取ると良いか説明する．そして分散処理フレームワークに足りない報告の受け取り側を整備し，実際に新聞記事検索システムにどのように応用するかについて説明する．この新聞記事検索システムは，GUI を使って途中報告を上手にユーザに示し，ユーザが入手した情報に満足した時点で検索を打ち切る機能を有している．

2 途中報告の意義と適用問題

コンピュータネットワークの発達に伴って，様々な用途で分散処理が行われるようになった．一般的な分散処理では，それぞれ分散した環境で任された仕事を行い，その仕事がすべて完了してから報告を行う形式をとる．しかし，実際には実行終了後に報告を行うのではなく，途中報告を頻繁に行うことが有効である問題が存在する．そのような問題の一つに，今回の新聞記事検索がある．

新聞記事検索などでは，欲しい情報を手に入れるためには膨大なデータを検索する必要があり，複数の記事が候補として得られる．我々は新聞記事検索のような検索を，

- 検索範囲が実効的に無限大
- 検索結果は複数
- 途中報告も有効

という検索問題にモデル化している．このような検索問題を考えたとき，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る．

新聞記事のリソースはもちろん有限である．しかし，出版社や年度などによっていくらでも追加が可能である．つまり，ある新聞記事リソースを検索し終ったとしても，別の新聞記事リソースを探しに行くとすれば，実効的にいつまでも探し続けることができる．また新聞記事検索では，ユーザの入力に対していくつかの候補記事が得られるので，検索結果は複数得られることになる．あいまいな入力を許すようなコンセプトの検索については，その性質上唯一の解というものは存在せず，類似度が高い解が必ずしもユーザが求めている解であるとは限らないのである．

本フレームワークの特長は，途中報告をすることにある．途中報告フレームワークの利点は，ある時点において，最適ではないかもしれないが，ある程度の解が求まることにある．そして途中報告された解によっては，現在実行中の処理の変更や中断を行なわなければならない．新聞記事検索では，ユーザは報告されてくる記事を閲覧し，自分の要求を満足する記事を発見した時点で，検索を終了させるであろう．

3 新聞記事検索システム

分散システムフレームワークのプロトタイプとして，新聞記事検索システムを作成した．本システムでは，新聞記事アーカイブを分割し，その各アーカイブを別々のホストに分散させてある．各ホストに繁説エージェントが移動し，そこで新聞記事検索プログラムを実行する．そして得られた解は検索がすべて終了するのを待たずに報告される．

検索アルゴリズムには，DP マッチングを採用している．DP マッチングを用いた検索 (文献[1]) の特長は，あいまいな入力を許し，検索対象となるリソースも生データに近いものが利用できる点である．しかし，柔軟な入力を許す反面，計

算量が多く，過去数年分の新聞記事を検索するというような場合には，多くの時間を要する．検索に多くの時間がかかったとしても，本システムでは途中報告を行なうので，ユーザは検索がすべて終了するまで待たないですむという特長がある．

3.1 システムの動作

新聞記事検索システムのGUIを図1に示す．ユーザはこのGUIを操作し，新聞記事の検索を行う．上部エリアでシステムの操作とqueryの編集をし，下部エリアで途中報告の閲覧をすることになる．

下部エリアでは，ソートされた途中結果が閲覧できる．本の絵をしたアイコンは変化通知オブジェクトである．変化通知オブジェクトは，常に最新のランキングが閲覧できるように，ランキングの変化をユーザに知らせる．あらかじめ上位何件までを注目するか設定しておくことができ，その上位ランキングに変動があったことを動くアイコンで通知する．また，新聞記事には削除ボタンが添えてあり，自分の要求する記事ではなかった場合は，削除することができる．

同時に複数のqueryで検索を行いたい場合は，新しいウィンドウを開いて行う．

基本的な検索手順は次のようになる．

1. queryの作成 (ファイルから読み込み)
2. 報告条件の変更
3. 検索の開始
4. 途中報告の読み込み，閲覧，削除
5. 検索の中断

queryは，IREX(情報検索コンテスト)の課題形式を用いている．queryを見て分かるように自由文入力であり，キーワードではない．従って，DPの特徴が必要な検索方法である．

報告条件というのは，ここでは途中報告される結果の最低ラインと途中報告をする間隔である．この報告条件に従って繁説エージェントは，途中報告をおこなうことになる．

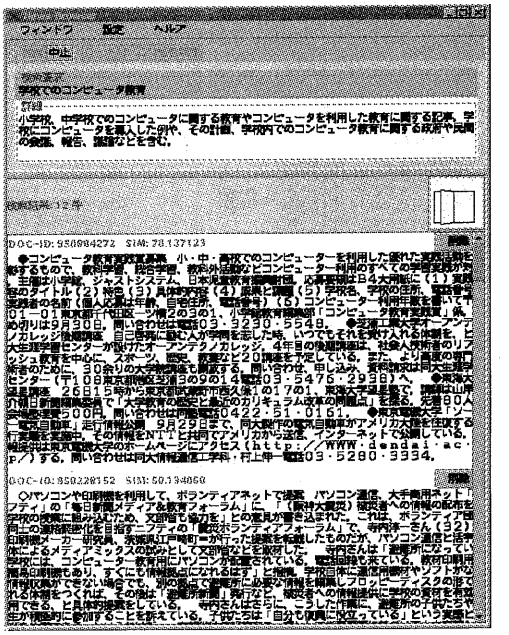

図 1: GUI

3.2 システム構成

新聞記事検索システムは図2のように構築される．各オブジェクトは本フレームワークの基本概念に従って作成されている．繁説エージェントは，分割・分散された新聞記事アーカイブが存在するホストへ移動し，新聞記事検索プログラムを実行する．そして報告条件に従って途中報告を行なう．

4 フレームワークの基本概念

本分散フレームワークの5つの基本概念を上げる．

1. ワーカー (Worker):
   ネットワーク上に存在する仕事場である．主な役割は，繁説エージェントを受け入れ，繁説エージェントから仕事を受け取り実行すること，繁説エージェントの移動の記録である．

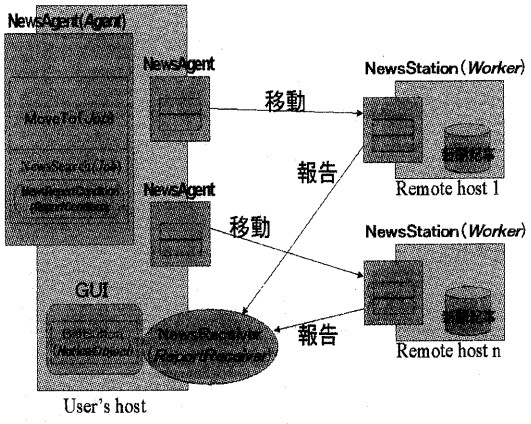

図 2: 構成図

2. ジョブ (Job):
本フレームワークでは，繁説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，ワーカーによって実行される．エージェントを別のワーカーに移動させる処理もジョブの一つである．

3. 繁説エージェント (Verbose Agent):
本フレームワークでは，分散させたい処理を繁説エージェントによって分散させる．繁説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．そして，ワーカーで実行されているジョブの途中結果を，報告条件に従って報告レシーバに報告する．

4. 報告条件 (ReportCondition):
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，途中報告を行なうタイミングや，ある条件を満たした結果のみを報告するための条件を設定する．この報告条件は，繁説エージェント (またはジョブ) に設定する．そしてワーカーに貯められている途中結果は，報告条件に従って報告されていく．

5. 報告レシーバ (ReportReceiver):
途中報告の受け取り側を整備するために必要となる．報告レシーバの役割は，途中報告を受け取る窓口となることと，途中報告を自動的に整理して保持し，途中報告に状態の変化を他のオブジェクトに知らせることである．報告レシーバに後処理を設定しておくことで，自動的に後処理を行ない，その変化を通知してくれる．また，報告レシーバを複数設置することで，異なった仕事に対する途中報告を個別に管理することができる．

6. 変化通知オブジェクト (NoticeObject):
ユーザに途中報告の変化を通知するためのオブジェクトである．得られる途中報告の中で，あらかじめユーザが注目する範囲を決めておき，その範囲に変化があったことをアイコンを変化させて知らせる．この概念は途中報告のフィルタの役割を果たし，ユーザに余分な情報を伝えない役割を持っている．

(1) から (4) までは文献 [2] で提案されたものであるが，我々はシステムを構築した経験から (5),(6) の追加を提案する．以下で (5),(6) の基本概念について詳しく説明する．

5 報告レシーバ

我々は途中報告を行なう分散フレームワークを設計したことで，分散処理が終了するのを待たずに有益な情報を得られるようになった．しかしシステムを構築する過程で，従来の単報告アプローチに比べ，途中報告アプローチは受け取り側の処理が複雑になることに気がついた．その主な問題点としては，後処理の呼び出しと処理内容の異なる報告を同時に受け取る場合が上げられる．我々は報告レシーバという概念を用いて，これらの問題点をフレームワークに従って処理している．報告レシーバは内部である仕事について途中結果を整理し，報告レシーバを複数個設置することで各仕事毎に途中結果を整理できる．

5.1 後処理の呼び出し

一般的に得られた途中結果には後処理を施すことが多い．新聞記事検索ならば後処理としてソートを行なうだろう．単報告アプローチの場合は報告がされた後に一度だけ後処理を行なうが，途中

報告アプローチでは後処理は何度も行なわなければならない．つまり後処理を非同期に何度もやり直さなければならず，またやり直しが可能なようにシステムを設計しなければならない．

しかし報告レシーバでは，後処理の内容をあらかじめ設定しておくことで，報告を得るたびに自動的に後処理を行なってくれる．つまり新聞記事検索では，プログラマが意識しなくとも，自動的にソートされた列を生成してくれるのである．もしも報告レシーバがなければ，システム作成者は途中報告を常に監視し，後処理を非同期に呼び出すようにプログラムを書かなければならないので大変である．

ところで報告レシーバは自動的に後処理を実行するので，プログラマは途中結果の列が変更されたことを知る手段が必要となる．そこで報告レシーバでは分散処理の状態や途中結果の状態を他のオブジェクトに知らせる機能を有している．一般的には報告レシーバからの通知を変化通知オブジェクトを通してユーザに通知する手段がとられることになる．このことは第6節で詳しく述べる．

5.2 異なる仕事からの途中報告問題

上述したように，報告レシーバの役割の一つは自動的に途中報告を整理することである．さらにもう一つ重要な役割に途中報告を仕事毎に整理する役割がある．

異なる仕事に関する途中結果が同じ場所に報告されると，途中報告が混ざってしまいその後の処理が複雑になる．この場合，それぞれの途中結果について後処理を行なうときは，プログラマはわざわざ途中報告をどの仕事についての報告か調べて処理を施さなければならないので，非常にプログラムが書きにくい．

同時に異なる種類の途中報告を受ける場合には，仕事の数だけ報告レシーバを設けるようにする．そうすることで，仕事毎に途中報告が受けることができ，途中報告を一括で制御する場合に比べて，その後の処理が容易になる．

今回の新聞記事検索システムでは，一つの query に対して一つの仕事と考え，query 毎に報告レシーバを設置して処理している．

6 変化通知オブジェクト

途中報告を受け取ったという情報をプログラム(またはユーザ)に知らせることは，重要である．なぜなら，もしも変化の通知が行われないと，たとえ有益な情報が早い段階で報告されたとしても，その情報を手にしていること気づかないからである．

本フレームワークでは，通知の流れは以下のようになる．

報告レシーバ
↓
変化通知オブジェクト
↓
人間
↓
GUI に表示

報告レシーバはプログラムに変化の通知を行う機能を持っている．そして報告レシーバから変化の通知を受けた変化通知オブジェクトは，実際にアイコンを変化させてユーザに変化を通知する．その変化通知オブジェクトによって，ユーザは初めて新しい報告を得たことを知り，表示の手続きを行なえるのである．

変化通知オブジェクトの重要な役割は，ユーザにただ変化を通知するだけではなく，余分な通知を行なわないようにフィルタリングすることである．これは，得られた途中結果の中で，ユーザが最も注目する範囲にあらかじめ決めておき，その範囲に変化があったときだけユーザに知らせるしくみである．

今回の新聞検索システムでは，報告レシーバが変化通知オブジェクトに知らせる変化の種類は6つある．

- 検索の開始
- 検索の終了
- 途中報告の入手
- ランキング変化
- 途中報告の削除

この中で，変化通知オブジェクトからユーザに知らされる情報は，検索の開始，終了，ランキング

変化の3つである．ランキング変化というのは上位何件かのランキングに変化があった場合に通知される変化である．これは，あらかじめ報告レシーバにソートの種類(スコア，日付)と注目する範囲(上位10件までなど)を設定しておくことで実現している．このランキング変化が通知された場合，変化通知オブジェクトはグラフィカルな形で人間に注意を促す．このようにすることで，ユーザは自分の注目する範囲に変化があったことを知ることができ，新しく入ってきた途中報告を表示させて閲覧することができる．

6.1 コーディング具体例

図3は，繁説エージェントに報告条件と報告先を設定し，ジョブを搭載してリモートホストへ移動するまでのコードである．localとremoteは，それぞれローカルホスト，リモートホストを示す．

1-3行目で途中結果を受け取る窓口となる報告レシーバを生成している．複数の異なった仕事を分散させて行なう場合には，仕事の数だけ報告レシーバを生成することになる．5行目で繁説エージェントを生成している．localとportは，それぞれ報告レシーバのホスト名とポート番号である．6,7行目で繁説エージェントにリモートホストへ移動する仕事と検索を行う仕事を格納している．8,9行目で検索条件をエージェントに設定している．10行目は一つ目の仕事を取り出して実行している．一つ目の仕事は移動ジョブ(MoveTo)なので，繁説エージェントはリモートホストへと移動する．

```
 1:  ReportReceiver receiver
= new ReportReceiver(this);
 2:  Thread thread = new Thread(receiver);
 3:  thread.start();
 4:  int port = receiver.getPort();
 5:  Agent agent = new Agent(local, port);
 6:  agent.add(new MoveTo(remote));
 7:  agent.add(new NewsSearch(query));
 8:  ReportCondition cond
= new ReportCondition(30000, 0.01);
 9:  agent.setReportCondition(cond);
10:  agent.go();
```

図3: サンプルコード

7 まとめ

繁説エージェントを用いた分散フレームワークに従ってシステムを実際に作り，頻繁に報告される途中結果の受け取り側の処理を備えた．途中報告フレームワークを提案した．実際に新聞検索システムを作り，その有効性を確認した．

今回，途中報告フレームワークは新聞記事検索に応用した．この新聞記事検索問題は，検索範囲が無限大，検索によって得られる解は複数，検索の途中結果も有効という3つの条件を満たす検索問題である．この問題については本フレームワークは有効である．我々は，この途中報告フレームワークは検索問題だけにとどまらず，いろいろな分散処理にも応用することを考えている．例えば，計算機負荷監視システムへの応用などが考えられる．分散処理に時間がかかり，その途中結果が有効である場合は，本フレームワークは有効であると考える．

参考文献

[1] 山本英子, 梅村恭司：日本語の技術文書における技術用語に着目したダイナミックプログラミングでの検索方法，情報処理学会プログラミングシンポジウム， pp131-142, 2000-1

[2] 山本英雄, 梅村恭司：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，情報処理学会研究報告 99-OS-81， pp131-136， 1999

[3] 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.

[4] 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.

[5] Eun-Seok LEE, Roberto Okada, Norio Shiratori: Agent-based Social Information Gathering on Internet, ICMAS-96, p448, 1996.