見込み計算を用いたニュースリーダの応答性改善法

池口祐子[†] 村山和宏[†] 上田和紀[††]

現在使われている会話システムの多くは，基本的に逐次処理パラダイムに基づいて記述してあり，コマンドの負荷が重い場合，コマンド発行からシステム応答まで時間がかかる．そこで，(1) ユーザのコマンド発行とシステムとの並行並列処理，(2) 漸増的計算，(3) 利用可能な計算資源を考慮に入れて計算する資源依存計算，といった要素技術に基づいて，会話システムの応答性改善を行った．本論文では，会話システムの応答性改善の実験として，ニュースリーダを題材とし，定量的に評価する．

Improving the Response of News Readers Using Speculative Computation

YUKO IKEGUCHI,[†] KAZUHIRO MURAYAMA[†] and KAZUNORI UEDA[††]

Most interactive systems, written as deterministic sequential programs, respond poorly to time-consuming commands. This paper discusses how we can reduce the responce (latency) of interactive systems using (i) concurrent/parallel processing between systems and user tasks, which is a form of speculative computation, (ii) incremental computation, and (iii) resource-dependent computation including real-time processing. The techniques were tailored and applied to our news reader composed of DNAS and GNUS. We have reimplemented some of the time-consuming GNUS commands, and the results of those initial experiments are reported.

1. はじめに

ネットニュースやWWW（World Wide Web）などネットワーク上の情報にアクセスするための会話システムは，ますます重要性が増しており，高度な機能を提供するようになっている．しかし，それらの機能の中には，多大な計算量を要するために応答性が悪いものがある．また，ネットワーク通信の遅延や輻輳のために，良好な応答性を確保できない場合が多いという問題もある．

そこで，本論文では，ネットニュースを題材として，その応答性を確保するための要素技術を検討し，実装，評価した[1)]．

要素技術としては，以下に述べるように，ネットワーク情報を利用したアクティブキャッシュと見込み計算（speculative computation）とが基本的な役割を担っている．アクティブキャッシュとは集団的なニュースリーダ利用情報から読まれる可能性が高いニュースグループを抽出し，そのニュースグループの未読記事をプリフェッチすることであり，これでキャッシュのヒット率が向上できる．また，見込み計算とは次に選択する可能性が高いニュースグループを抽出し，そのヘッダリストを起動時およびユーザのコマンド入力がない間に準備することであり，これで未読記事数が多い場合，ニュースグループ選択から記事購読までの時間を短縮することができる．しかもこれらは相互に関連し，アクティブキャッシュを行うことで見込み計算によるネットワーク負荷を最小限におさえることができている．

ネットニュースを題材としたのは，ニュースリーダ利用者の挙動がかなり正確に予測でき，アクティブキャッシュと見込み計算が有効に機能すると期待できるからである．最近では，ネットワークにおけるWWWの比重が急増しているが，ネットニュースはWWWとは存在意義も利用形態も異なり，その応答性を改善する必要性は大きい．

会話システムの設計目標は，ユーザとシステムの円滑な会話（＝通信）を実現することである．その方法として急速に発展しつつあるのが，マルチメディア技術等による会話のバンド幅の改善であるが，現実には，

† 早稲田大学大学院理工学研究科情報科学専攻
Major in Information and Computer Science, Graduate School of Science and Engineering, Waseda University
†† 早稲田大学理工学部情報学科
Department of Information and Computer Science, Waseda University

マルチメディア化にともなって応答性を犠牲にしていることが多い．応答性は会話の質を決めるもう1つの重要な要素であり，これを改善する方法論を確立することは，円滑な会話を実現するためにきわめて重要である．

以下，2章では，応答性改善のための要素技術を述べ，3章では，2章で述べた要素技術に基づき，ニュースリーダの応答性改善法を提案する．4章では，実装したニュースリーダを定量的に評価する．

2. 応答性改善のための要素技術

会話システムとの円滑な会話を実現するには，システムがユーザの要求に対して十分な速さで反応することが重要である．必要な応答速度については様々な研究が行われている[9]が，ほとんど計算時間を要しないものでは，人間同士の会話における反応時間（0.2秒程度）と同程度の応答性を確保する必要があり，かなり計算時間がかかる要求でも約2秒以内には何らかの反応を返すことが望ましい．また，その反応内容は，“お待ちください”などよりも処理結果の一部の方が望ましい．この要求を満たすには，次のような技術が必要となる．

(1) **見込み計算**（**投機的計算**，speculative computation）

会話システムを1つのプロセス，ユーザの一連の作業をもう1つのプロセスと見なすと，両者は互いに通信しあう並行システム（concurrent system）を構成する．しかし，通常の会話システムは，決定的な逐次プログラムとして構築されており，ユーザの入力を待つ時間を有効に活用していない．ユーザの入力を受けて初めて作業を開始したのでは，処理時間のかかる要求に敏速に反応することができない場合がある．そこで，会話システムでは，以下に述べる見込み計算の技術が重要となる．

見込み計算とは，後で役立つことを期待して，必要性が確定する前に行う計算のことである．見込み計算の意味を広くとらえると，ハードウェアによる分岐予測や入出力データのバッファリングなど，計算機システム内で早くから利用されてきた．最近では並列計算環境の普及にともない，システムプログラムまたはアプリケーションレベルでも，どのような投機が有効かを検討し，プログラムすることが重要になってきている[5),6)]．

見込み計算に共通する目的は，何らかの処理要求に対する応答性改善にあるので，会話システムの場合は，逐次計算環境でも重要な要素技術となる．並列計算環境での会話システムにおける見込み計算は，文献8)で論じられているが，会話システムにおける見込み計算の意義は，並列計算環境の有無とは独立であると考えられる．

見込み計算の的中率をあげるには，過去のユーザの行動や見込み計算の成否を統計的に解析して役立てることが重要である．しかし，会話システムにおける見込み計算は，後述するように，的中率を最優先に行うべきであるとは限らない．的中時の応答時間の短縮効果も考慮する必要がある．的中率が低くても応答性改善効果の高い処理を，的中率が高いが応答性改善効果が低い処理に優先して行うことが望ましい．

見込み計算に利用できる時間は，会話システムの入力待ち時間だけではない．ネットニュースのように定期的に起動されるシステムでは，ユーザが会話システムを起動していない時間帯も有効活用できる．

(2) **漸増的計算**（incremental computation）

漸増的計算とは，計算結果を最後に一括して返すのでなく，できた部分から順次返す計算のことである．これは，会話システムにおいて2つの意義を持つ．ユーザの待ち時間を短縮できることと，システムの計算，表示作業とその表示に対するユーザの作業を並列化できることである．

たとえば，**図1**のような2つのアルゴリズムA，Bを仮定する．総計算時間は，アルゴリズムAの方がアルゴリズムBより短い．しかし，アルゴリズムAが結果をすべて求めた後で表示するのに対し，アルゴリズムBは結果をできた順に表示する．図に示すようなユーザの作業速度を仮定すると，ユーザが作業を開始できるまでの時間，作業が終了するまでの時間のいずれをとっても，アルゴリズムBの方が優れている．

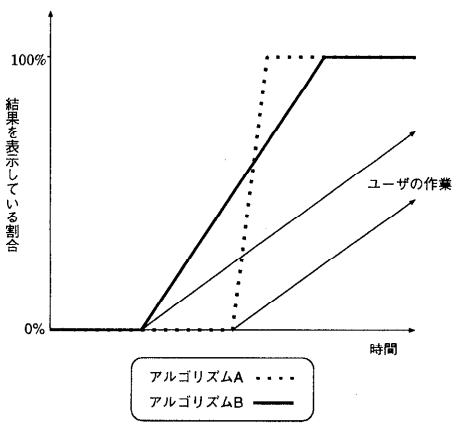


図1 漸増的計算の例
Fig. 1 An example of incremental computation.

会話システムでは，ユーザの要求に対する処理の計算量が大きく，計算時間がかかることが少なくない．このような場合に漸増的計算は効果的である．待ち時間と総計算時間の双方を最適化するためには，総計算時間の短いアルゴリズムAと漸増的に結果出力が可能なアルゴリズムBを並行動作させ，アルゴリズムAの終了までアルゴリズムBの結果を表示し，その後アルゴリズムAの結果を表示することが考えられる．

(3) **資源依存計算**（resource-dependent computation）

資源依存計算とは，その時点で利用可能な計算資源を考慮して行う計算のことである．ここでの計算資源は，ネットワーク負荷，計算機負荷（ロードアベレージ），制限時間，ディスク容量などがあげられる．

資源依存計算は，見込み計算を行う場合特に重要になる．本論文で仮定している計算機環境は，比較的計算機負荷に余裕がある場合を想定しているが，マルチタスク，マルチユーザ環境の場合，見込み計算によって計算機負荷をいたずらに大きくして，他の作業の応答性を悪くしてはならない．いたずらにディスク容量を使うのも望ましくない．また対外的には，ネットワーク負荷のいたずらな増加は避ける必要がある．そこで，会話システムは，外部ネットワークや計算機の負荷が重い場合，応答性改善のための見込み計算は行わず，負荷が下がったときに再開するようにする．また，見込み計算の結果を格納する記憶容量が事前に決められた上限に達した場合，それ以上の見込み計算は行わない．

これらの要素技術は，計算機科学の特定の分野ですでに研究，利用されているが，プログラミングの考え方や技術や道具がまだ十分普及していない．

3. 要素技術のニュースリーダへの応用

本章では，会話システムの例としてニュースリーダを取り上げ，2章で述べた要素技術の実装法を提案する．本論文におけるニュースリーダとは，外部ネットワーク上に分散している複数のニュースシステムへの同時アクセス機能を提供しているDNAS（Distributed NNTP servers Access Service）[7]とGNU Emacs上で使用されるニュースリーダGNUSを組み合わせたものであり，ユーザがGNUSでDNASにアクセスすることにより，複数のニュースシステムで管理されているニュースグループを同時に読むことができる．

3.1 現在のネットニュースの状態とその購読傾向

まず，ニュースリーダの応答性改善のために，ネットニュースの現状とその購読傾向について述べる．

ネットニュースの現状および購読状況として，次のようなことがあげられる．

(1) 有効なニュースグループ数が1万個を超え，現在も増加している．

(2) ニュースグループは，各個人ごとに，次のように分類できる．
- (a) 毎回，ほぼ確実に読むニュースグループ
- (b) 毎回読むとは限らないが，記事がエキスパイアされるまでにはほぼ確実に読むニュースグループ
- (c) 時間に余裕のある場合に目を通すニュースグループ
- (d) subscribeしているが，ほとんど目を通さないニュースグループ

(3) ネットニュースの記事はバケツリレー方式で各ニュースサーバに伝播していくため，投稿された記事がすぐ読めるとは限らない．

次に，現在有効なニュースグループから無作為に抽出した約30個のニュースグループに関する，購読率を使用したネットニュース購読状況を図にしたものが**図2**である．購読率は，あるユーザがあるニュースグループの記事をどの程度読んでいるかを表す指標で，購読率＝（既読記事数/現在有効な総記事数）で計算される．これを利用することで，総記事数が違うニュースグループの興味度を一律に表すことができる．この図から分かることは次のとおりである．

(4) 同じ計算機システムを共有する人々が興味を持つニュースグループは共通しているものもあるが，そうでないものに関しては，興味のばらつきが大きい．

3.2 要素技術の実装法

ここではニュースリーダにおける2章で述べた要素技術の実装法を述べる．

3.2.1 DNASへのアクティブキャッシュの採用

DNASは，GNUSと複数のニュースシステムの仲介をしているが，第3版からは，要求のあった記事を提供すると同時にその記事をキャッシュし，同じ記事が再び要求された場合，外部ネットワーク経由で調達せずにそのキャッシュの内容を返すという機能が追加された．

このキャッシュ機能の長所は，ある記事が複数回要求された場合，キャッシュにヒットし，外部ネットワーク負荷を緩和できるだけでなく，応答性改善にもつながることである．

しかし，このキャッシュ機能は，必要になった記事を

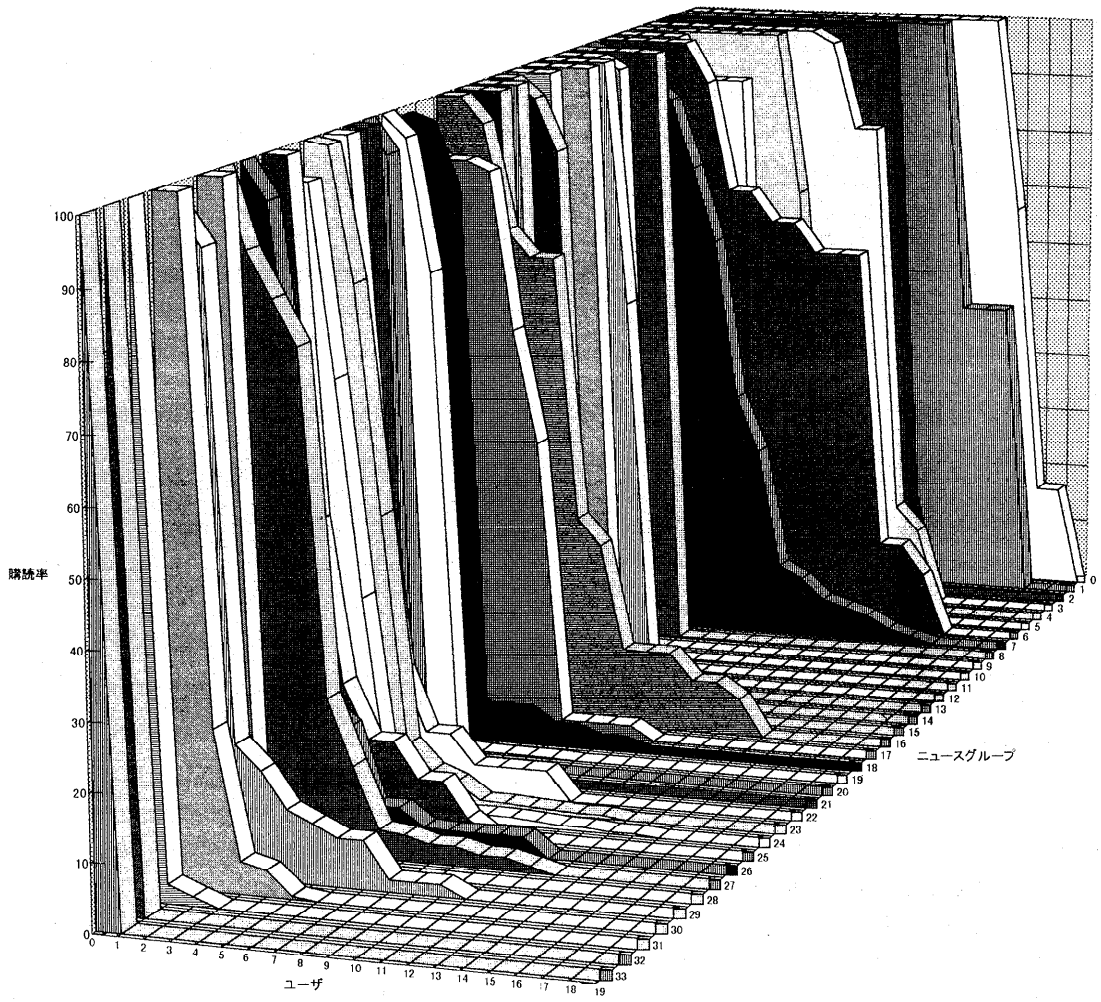


図2 無作為抽出したニュースグループの購読状況
Fig. 2 The reading rates of randomly chosen newsgroups.

外部から調達した時点で同時にキャッシュし，一定期間アクセスがないとエキスパイアするというパッシブキャッシュを採用しており，次のような問題点がある．

(1) 複数回アクセスされる記事にしか効果がない．
(2) 読まれる可能性を考慮せずに機械的にキャッシュするので，キャッシュのヒット率は上がらない．
(3) キャッシュしておく必要がなくなった記事は，エキスパイアされるまで無駄にキャッシュに残ってしまう．

キャッシュのヒット率は高ければ高いほど，外部ネットワーク負荷を緩和でき，かつ応答性が改善される．また，無駄なキャッシュを減らせば減らすほど，ディスク領域を有効活用することができる．

そこで，ユーザが読む可能性が高い記事を事前にキャッシュするというアクティブキャッシュを提案する．アクティブキャッシュは，読まれる可能性が高い記事を予測して事前にキャッシュするため，1回目のアクセスからヒットする．また，その記事を読むユーザを予測してエキスパイアを制御するため，無駄なキャッシュは最小限におさえられる．よって，パッシブキャッシュよりもヒット率が高く，記事の再使用回数も多くなることが期待できる．

アクティブキャッシュは一種の見込み計算であり，外部ネットワークの通信総量と計算機負荷はわずかに増加するが，通信の時間帯を夜間などの低負荷時へずらすことで，会話システムの利用時の負荷を小さくできる．

3.2.2 GNUSにおける起動時の応答性改善

従来のGNUSでは，起動時にニュースシステムと接続し，未読記事番号リスト（ニュースグループにお

ける現在有効な記事番号から既読記事番号を除いた記事番号リスト）をすべて作成した後，ニュースグループ選択画面を出力する．そして，この画面でニュースグループを選択し，記事を購読する．現在，ニュースグループは1万個を超えており，すべての未読記事番号リストの作成には時間がかなりかかる．これが，起動からニュースグループ選択までの応答性低下の原因である．また，ユーザは興味があるニュースグループから順に記事を読むため，起動時にすべての未読記事番号リストを用意する必要はない．

そこで，未読記事番号リストを漸増的に作成，表示することで早期に記事を読み始められるようにし，残りの未読記事番号リストはユーザが記事を読んでいる間に作成する．残りの未読記事番号リストの作成はユーザのコマンド入力がない間に漸増的に行い，ユーザの要求に対する応答性低下を防ぐ．

3.2.3 GNUSにおけるニュースグループ選択時の応答性改善

従来のGNUSでは，ニュースグループ選択時にその未読記事のヘッダリストをサーバから取得し，記事の親子関係（スレッド）を解析した後，スレッドを反映したヘッダリストを記事選択画面に出力する．ニュースシステムへの1コマンドに対して1個の記事のヘッダしか転送されず，そのヘッダに対し，Message-IDの切出しなどのテキスト処理を逐次的に施すため，未読記事が多い場合，ヘッダリストの作成には時間がかなりかかる．これが，ニュースグループ選択から記事購読までの応答性低下の原因である．

そこで，次に選択する可能性の高いニュースグループのヘッダリストの準備を，見込み計算によって，起動時およびユーザのコマンド入力がない間に行う．

的中率の観点からは，3.1節の（2）の（a）に属する記事の準備が最も確実性が高い見込み計算である．しかし，（2）の（a）に属するニュースグループの未読記事数は比較的少数個であることが多く，見込み計算のメリットは大きいとは限らない．一方，（2）の（b）に属する記事が現在のセッションで読まれる可能性は必ずしも高くないが，（2）の（b）のニュースグループには多くの未読記事がある場合が少なくない．それ自身がそれらのニュースグループの選択を後回しにする一因にもなる．よって，そのようなニュースグループが選択されたときの応答性を改善する意義は大きい．

4. 実装内容と評価

4.1 アクティブキャッシュシステム

ここでは，アクティブキャッシュシステムの基本設計と，キャッシュやエキスパイアの対象となる記事を決定するアルゴリズムについて述べる．

4.1.1 システムの基本設計

3.2.1項で述べたアクティブキャッシュをDNAS第3版に外付けした．この結果，DNAS第3版に付属しているパッシブキャッシュ機能に代わり，DNASは記事が要求されると，その記事をアクティブキャッシュがキャッシュしていればキャッシュから，なければ外部ネットワーク経由で調達する．

アクティブキャッシュを実現する場合，キャッシュ対象の単位として，個別記事単位，ニュースグループ単位が考えられるが，現在，有効なニュースグループ数は1万個を超えているので，ニュースグループ単位で判定するのが最も現実的である．そこで，キャッシュ対象はニュースグループ単位で判定し，キャッシュ対象になったニュースグループは，有効なすべての記事をキャッシュすることにした．これに対し，エキスパイアは，キャッシュに存在する個々の記事の購読状況に基づき，記事単位で行うことにした．

本論文では，購読率の高い人が多く存在するニュースグループを優先し，事前に決めたディスク容量に達するまでキャッシュをすることにした．キャッシュ間隔は1時間だが，その時点での平均計算機負荷が高い場合，15分ごとに計算機負荷を調べ，負荷が小さくなるまでキャッシュを行わない．

購読率は，1週間ごとに自動的に各ユーザの.newsrcという既読記事番号リストから測定し，キャッシュ順位を決定する．このような新たなデータを蓄積するリストを使用することで，1週間に1度もネットニュースを読まなくても，先週までの購読傾向をキャッシュ順位に反映できる．

4.1.2 キャッシュとエキスパイアを行うためのアルゴリズム

次に，キャッシュとエキスパイアのアルゴリズムについて述べる．

図2の中から，あるニュースグループの詳細な購読結果を**図3**に示した．

ここで，キャッシュに存在するのに読まれない危険率の和 α とキャッシュからエキスパイアされた後で読まれる危険率の和 β を考える．購読率の平均値（図中の破線）でエキスパイアするとしたとき，この2つの危険率の和を図示すると図3になる．危険率の和 α

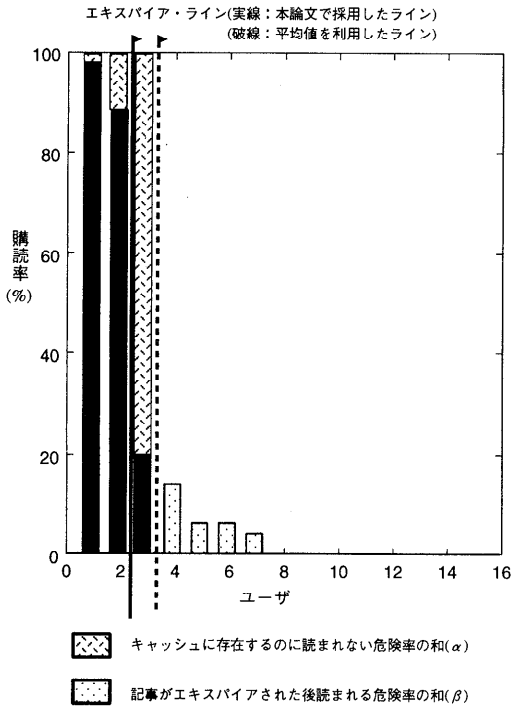


図3 ニュースグループにおける購読結果
Fig. 3 A typical newsgroup reading characteristic.

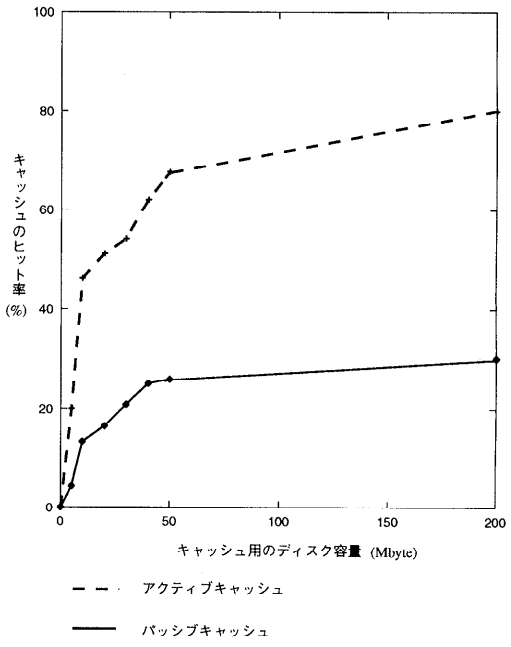


図4 キャッシュの実験結果
Fig. 4 The results of active caching and passive caching.

と危険率の和 β のそれぞれの重要度は，キャッシュのディスク容量に関係する．

キャッシュのディスク容量が少ないとき ディスクを効率良く使用するため，危険率の和 β よりも危険率の和 α を重要視し，エキスパイアを早めに行う．

キャッシュのディスク容量が多いとき キャッシュのヒット率を高めるため，危険率の和 α よりも危険率の和 β を重要視し，エキスパイアは遅めに行う．

しかし，用意できるディスク容量によって α と β のどちらかの重要度を大きくすると，購読率の高い人と低い人の差が激しいというネットニュース購読の特徴から，重要度の低い方の危険率の和が極端に高くなってしまう．そこで，この特徴を利用し，購読率の高い人と低い人の境界線でエキスパイアする．そうすると，α に関係する購読率の高い人はそのニュースグループを読む可能性が非常に高いので α を低くおさえることができ，また β に関係する購読率の低い人はそのニュースグループを読む可能性が非常に低いので β を低くおさえることができる．図3の場合は，図中の実線でエキスパイアを行う．ここで，単なる人数ではなく，実線の左側の高購読率のユーザがすべて読んだかに基づくことによって，より厳密なエキスパイアができる．

一方，キャッシュは，最大購読率が高いユーザがいるニュースグループを優先的に行う．しかし，同程度の最大購読率を持つニュースグループが複数存在した場合，記事の再ヒット率を考慮し，最大購読率に近い購読率のユーザが多いニュースグループを優先する．

このようなキャッシュとエキスパイアのアルゴリズムを使用することで，1人でも興味度が高い読者がいるニュースグループは早めにキャッシュされ，読む可能性のある読者全員が読んだ記事はただちにエキスパイアされるという効率の良いディスクの使用を実現できる．

4.2 実験結果と考察

アクティブキャッシュとパッシブキャッシュにおける実験結果を，図4に示す．ただし，パッシブキャッシュのヒット率は，そのディスク容量に達したときに測定した．実験に用いたDNASのユーザ数は12名であった．

アクティブキャッシュのために生成されたキャッシュ順位表によると，実験期間中に1人でも読んだことがあるニュースグループは554個であった．そのうち，複数の読者がいたニュースグループは1/3であった．

パッシブキャッシュは，ユーザが1度読んだ記事はすべてキャッシュするので，読者が1人だけのニュース

グループは，キャッシュしても読まれない可能性が高い．しかも，ニュース購読状況によれば，複数の読者がいるニュースグループは全体の1/3にすぎず，2/3のニュースグループの記事が読まれずにキャッシュに残ってしまう．よって，パッシブキャッシュでは，ディスク容量が増加しても，読まれない記事も増加するため，ヒット率はそれほど増加しない．これに対し，アクティブキャッシュは，読まれる可能性の高い記事をプリフェッチし，読まれる可能性がなくなればただちにエキスパイアするので，ディスク容量を効率良く使用できる．よって，同じディスク容量でもパッシブキャッシュよりもヒット率が高くなる．

本実験環境では，200 Mbyteで読者が1人でもいるニュースグループはすべてキャッシュできるので，それ以上のディスク容量は必要ないが，ディスク容量を200 Mbyteにしても，ヒット率が100%になるわけではない．これは今まで購読率が0%のニュースグループが読まれたり，エキスパイアされた記事が読まれたりしたためである．

また，ディスク容量が50 Mbyteに達するまではキャッシュのヒット率は順調に増加しているが，その後はそれほど増加しない．これは，50 Mbyteで3.1節の（2）の（a）と（b）に該当するニュースグループはほとんどすべてキャッシュしたため，50 Mbyteから200 Mbyteまでは（c）に該当するニュースグループをキャッシュすることになり，これは読む可能性が低いからである．

4.3 GNUSの応答性改善

ここでは，3.2.2項および3.2.3項で述べた応答性改善法に従い，改良GNUSの基本設計と実験結果および考察について述べる．

4.3.1 システムの基本設計

改良GNUSの基本的な挙動は，最初にいくつかのニュースグループの未読記事番号リストを作成して，それらをニュースグループ選択画面に出力する．その後，漸増的に残りの未読記事番号リスト作成とヘッダリスト作成を並行して行い，できた未読記事番号リストからニュースグループ選択画面に出力していくというものである．ヘッダリストを作成したニュースグループは，ニュースグループ選択画面上で該当するニュースグループの先頭にマークをつけることで，作成したことを表示するようにした．よって，マークがついたニュースグループを選択すると，ただちに記事選択画面が表示される．

未読記事番号リストを作成する際，ユーザが選択する可能性が高いニュースグループを優先的に作成することが重要である．そこで，過去に読んだことがあるニュースグループをユーザが選択する可能性が高いと考え，GNUS起動中にとったニュースグループ選択の履歴から，優先的に作成するニュースグループを決定している．

また，ヘッダリスト作成の場合，一番効果の大きいと思われる3.1節の（2）の（b）に属するニュースグループを優先的に行う．具体的には，ニュースグループ選択の履歴の中で未読記事数が大きいもののみをヘッダリスト作成の対象にしている．

漸増的計算は，ユーザの作業を邪魔せずに行う必要がある．よって，ユーザの作業と計算機の作業を並行化するために，ユーザからのキーボード入力があれば，その時点の計算機の作業を中断して，ユーザの要求に対する処理を優先的に行う割込み処理を行えばよいが，GNUSの記述言語であるEmacs Lispには割込み処理がない．そこで，キーボード入力を監視し，ある一定時間キーボード入力がない場合，計算機は次のキーボード入力の邪魔にならない程度に小さく分割した作業を実行するようにした．

本論文では，ユーザが1秒以上キーボード入力を休んだ後のキーボード入力への応答は1秒以内に行えばよいものとし，1秒間キーボード入力がなかった場合，1秒以内に終了するように分割した作業を実行するようにした．具体的には，ニュースグループの未読記事番号リスト作成をこの方法で行っている．

しかし，1秒以内に終了するように容易に分割できない作業も存在する．そのような作業はGNUS上で行わず，外部プロセスを生成することで，ユーザの作業との並行化を実現した．具体的には各ニュースグループのヘッダリスト作成をこの方法で行っている．ただし，外部プロセスを生成する方法は，プロセスの生成と結果の回収に時間がかかり，Emacs Lispのデータに自由にアクセスできないので，粒度の細かな漸増的計算には適さない．

これらの技術を踏まえ，改良GNUSの挙動を時間軸に沿って表した図が**図5**である．

4.3.2 実験結果と考察

実験はSun Sparcstation 20で行い，計測はGNUSのプログラム内に時間を測定する関数をうめて行った．

GNUSの起動後，ニュースグループ選択画面を初めて出力するまでの時間を従来のGNUSと改良GNUSで比較したものが**図6**である．総時間では，従来のGNUSが48秒かかるのに対し，改良GNUSは7秒ですんでいる．

この応答性改善ができた理由には次のような技術が

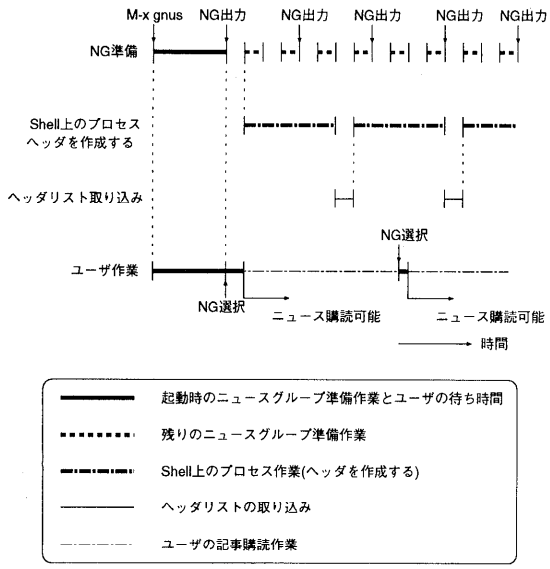


図5 改良GNUSの時間軸に沿った挙動
Fig. 5 The behavior of improved GNUS with time.

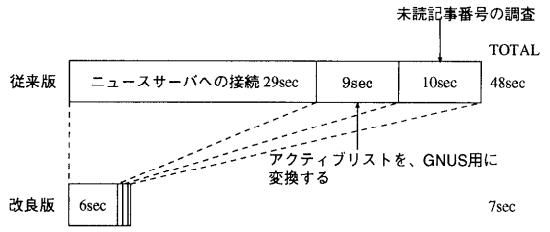


図6 GNUS起動後ニュースグループ選択画面を初めて出力するまでの時間
Fig. 6 Improvement of GNUS startup overhead.

あげられる.

- 最初にすべてのニュースグループの準備を行わず,読む可能性がある20個を選び,それだけの未読記事番号リストを作成してニュースグループ選択画面に表示した.残りの準備は記事を読んでいる間に漸増的に行う.
- 未読記事番号調査に必要なアクティブリストを外部ネットワーク経由で調達せず,DNASが1時間ごとにキャッシュしているリストを使用した.このリストはすべてのニュースグループの情報を含むため非常に容量が大きく,外部ネットワーク経由で調達するにはかなり時間がかかる.実際,従来のGNUSでは29秒かかるのに対し,改良GNUSでは6秒ですむ.欠点としては,最新の記事を読むことができない危険性があることだが,ネットニュースの特徴で,投稿された記事がすぐ読めるとは限らないことと,最新状態との差が最長でも

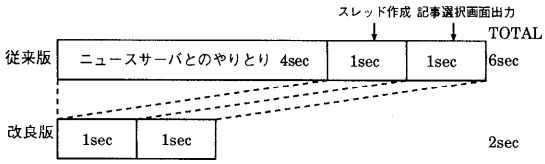


図7 ニュースグループ選択時にかかる時間
Fig. 7 Improvement of newsgroup selection overhead.

1時間以内であることから,問題はない.

起動時の応答性改善は大幅な効果が得られるが,すべてのニュースグループの準備が終わるまでにかかる時間は,従来のGNUSでは48秒であるのに対し,改良GNUSでは148秒もかかる.この原因は,改良GNUSが最初の20個のニュースグループをニュースグループ選択画面に出力した後,残りのニュースグループの準備とヘッダリストの作成をキーボード入力を監視しながら漸増的に行うためである.しかし,最初にニュースグループ選択画面を出力した後は,ユーザの作業と計算機の作業が並行化できるため計算機の残りの作業は隠蔽することができる.

次に,ニュースグループ選択時に,ヘッダリストが準備されていた場合とされていない場合の記事選択画面が出力されるまでの時間を比較したものが**図7**である.準備するヘッダ数は各ニュースグループごとに最新50個と一定にしている.これによって,未読記事数が極端に多いニュースグループのヘッダリスト作成に時間がとられないので,ヘッダリストを準備できるニュースグループ数が増し,ユーザの選択の幅が広がる.また,残りのヘッダリストは,用意した50個の記事を読んでいる間に,10個ずつ漸増的に作成する.

図7のとおり,ヘッダリストをあらかじめ準備することで,6秒から2秒へ応答性が改善されるので,興味はあるが記事選択画面出力まで時間がかかるので避けていたニュースグループを気軽に眺めることができる.

また,本方式は,今まで読んだことがあるニュースグループのうち,未読記事数が多いすべてのニュースグループのヘッダリストを作成しており,必ずしも準備したものすべてが読まれるとは限らない.しかし,DNASのアクティブキャッシュのヒット率は適当なディスク容量を与えれば50%を超えるので,外部ネットワーク経由で調達するヘッダは半数以下ですみ,いたずらにネットワーク負荷を増大させることはない.

5. おわりに

本論文では,負荷に余裕のある計算機システムで動く会話システムの応答性を,見込み計算によるユー

ザ・システム間の並行並列処理，漸増的計算，資源依存計算の要素技術に基づいて改善する方法について，ニュースリーダを題材として検討し，これまでの実験結果を報告した．

会話システムの設計・記述には，従来の決定的な逐次プログラミング・パラダイムでは不十分な場合がある．本論文で述べた各要素技術は，コンピュータアーキテクチャ，オペレーティングシステム，人工知能，並列処理などの各分野で研究されており，それらのパラダイムの流通，普及をはかる必要がある．

“ユーザを待たせないこと”は，良い会話システムには大切な条件である．そのためには，“ユーザが要求してから処理を開始する”という要求駆動処理から見込み計算のパラダイムへ移行し，その成熟を図る必要がある．また，計算結果を漸増的に出力することで，ユーザの待ち時間を隠蔽する技術も重要である．これには，漸増的出力が可能なアルゴリズムの採用以前に，コマンド仕様を漸増的出力が可能なように（再）設計するという観点も必要になる．

今後の課題としては次のようなことがある．

本論文で述べたアクティブキャッシュシステムでは，キャッシュとエキスパイアの対象を，.newsrcという各ユーザの既読記事番号リストから抽出した．このリストだけでは，どの記事を読んだかといった静的な情報は得られても，その記事をどういう順番で読んだかといった動的な情報は得られない．そこで，既読記事番号リストに加え，各ユーザのネットニュース購読行動も，キャッシュやエキスパイアの対象決定の要素に入れると，より精度の良い見込み計算ができることが期待できる．

本論文で述べた改良GNUSでは，GNUS起動時において，アクティブリストや既読記事リストの取得作業は今までどおり，すべてのニュースグループについて行っており，これらの作業に関しても最初に出力するニュースグループのみに対して行えばさらなる応答性改善が期待できる．

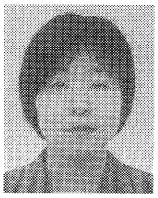
また，ニュースグループ選択時までに完了している見込み計算は，ヘッダリストの準備だけで，記事の親子関係を解析する作業はニュースグループ選択時に行っている．親子関係が複雑な場合，解析するのに時間がかかる場合がある．そこで，ユーザからのコマンド入力がない間に漸増的に親子関係の解析まで行えば，ニュースグループ選択時にかかる時間は記事選択画面にヘッダを出力する時間だけになり，さらなる応答性改善が期待できる．

謝辞 本研究にあたり，熱心に議論していただいた，早稲田大学理工学部情報学科上田研究室の学生諸氏に感謝します．特に，改良GNUSの初期の版の共同研究開発者であった旭哲男氏，ネットワークニュースシステム全般について有益な助言をいただいた松田慎一氏には深く感謝いたします．なお本研究の一部は，早稲田大学特定課題研究（96A-264）の成果によるものである．

参考文献

1) 旭 哲男，池口祐子，村山和宏，上田和紀：見込み計算を用いたネットワークニュースリーダの応答性改善法，日本ソフトウェア科学会WISS'95，インタラクティブシステムとソフトウェアIII，田中二郎（編），pp.113–122，近代科学社，東京(1995).
2) 旭 哲男：バックグラウンド計算を用いたGNUSの性能改善，早稲田大学理工学部卒業論文(1995).
3) 池口祐子：統計情報を用いたネットニュースにおけるキャッシュ機能の設計と実装，早稲田大学理工学部卒業論文(1995).
4) 村山和宏：見込み計算を用いたネットニュースリーダの応答性改善法，早稲田大学理工学部卒業論文(1996).
5) 上田和紀：理論と実際のギャップ：並列プログラミング，情報処理，Vol.34, No.7, pp.845–847(1993).
6) 上田和紀：並行プログラミングとGHC，続・新しいプログラミング・パラダイム，井田哲雄，田中二郎（編），pp.1–34，共立出版，東京(1990).
7) 梶田将司，今井祐二：DNASD3/README.jis，ftp://ftp.waseda.ac.jp/pub/news/DNAS/DNASD3.tar.gz
8) 舘村純一：投機的実行を応用したインタラクティブ並列プログラム，情報処理学会研究報告，Vol.95, No.82, pp.97–104 (1995).
9) B. シュナイダーマン（著），東 基衛，井関 治（訳）：ユーザインタフェースの設計，日経BP社，東京(1993).

(平成8年9月17日受付)
(平成9年4月 3 日採録)

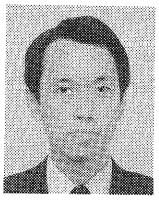


池口　祐子

1971年生．1995年早稲田大学理工学部情報学科卒業．1997年同大学院理工学研究科情報科学専攻修士課程修了．同年（株）東芝入社．現在（株）東芝情報通信・制御システム事業本部勤務．ネットワーク・プログラミング，ヒューマン・インタフェース，学習に興味を持つ．日本ソフトウェア科学会，人工知能学会各会員．

村山　和宏（学生会員）

1973年生．1996年早稲田大学理工学部情報学科卒業．現在同大学院理工学研究科情報科学専攻修士課程に在学中．ネットワークプログラミング，インタラクティブシステム，知識情報処理に興味を持つ．



上田　和紀（正会員）

1956年生．1978年東京大学工学部計数工学科卒業．1986年同大学院情報工学博士課程修了．工学博士．1983年日本電気（株）入社．1985～1992年（財）新世代コンピュータ技術開発機構へ出向．1993年早稲田大学理工学部情報学科助教授．1997年同教授．プログラミング言語の設計と実装，並行・並列処理，論理プログラミング，インタラクティブシステムなどの研究に従事．日本ソフトウェア科学会，人工知能学会，ACM，IEEE Computer Society，Association for Logic Programming各会員．Journal of Logic Programming言語・システム分野エディタ．