「*Internet*広域分散協調サーチロボットの研究開発」
研究成果報告書

村岡洋一
E-mail: muraoka@muraoka.info.waseda.ac.jp
学校法人　早稲田大学

概　要

世界中のWWWサーバから発信されるデータは2000年1月時点で約17.8億ページと予測される．そして，これらのWWWサーバのデータを自動的に収集するプログラムを一般にWWWロボットと呼ぶ．このWWWロボットが全WWWサーバにアクセスし，必要なWebページを収集するには，現状技術ではWWWロボットを動作させるサーバの性能および回線容量の制限などから，日本国内のみのWWWサーバを対象とした場合でも１ヶ月以上の時間を要する[15]．さらに，調査によればインターネットに与える負荷の約４分の１がWWWロボットのアクセスによって占められており，インターネットに与える影響としても無視できなくなってきている．

このような現状を打破するため，Webページをインターネット上に分散された複数のWWWロボットで協調して高速収集するための研究開発を実施した．これにより，従来20以上のWWWロボットが同一のWWWサーバにアクセスし同じデータを収集するという無駄を無くし，かつ，分散協調して集めることにより収集の高速化を目指した．

分散協調サーチロボット（ソフトウェア）の研究開発では，日本国内(ドメイン名がjpで終わるWWWサーバ)の全Webページを24時間以内に収集することを目標とした．

平成10年度は，「分散協調サーチロボットのプロトコルの検討」，「分散エリア決定方法の検討」，「分散協調サーチロボットプロトタイプシステムの設計と先行評価」を行い，２年目となる平成11年度では，「分散協調サーチロボットの開発」「分散協調サーチロボット環境の実装」，「分散協調サーチロボットの評価」を行った．

実際に，日本全国の33個所に分散協調サーチロボット(分散型WWWロボット)を配置し，日本国内のWWWサーバの内，50,000サーバを対象として収集実験を行った．ネットワークの動的変動等の影響のため，評価は１７ヶ所に分散配置された分散型WWWロボットを使い6,500のWWWサーバを対象として行い，一カ所で集中して収集する場合に比較し，負荷均一化による分散により，6.3～286倍の高速化が可能であることがわかった．

平成12年3月末には，jpドメインに存在する全WWWサーバのデータを24時間以内に収集するシステムとして実運用を開始する．

目次

1 はじめに

World Wide Web (WWW) は，Mosaic が開発されて以来，インターネット上での情報提供及び情報収集の手段として世界中で利用されている．特に，情報検索サービスは，今やインターネットのポータルサイトとして認知されるに到っている．しかし，以下に述べるように多くの問題が残されている．

ブラウザが世の中に認知されはじめた 1993 〜 1994 年頃には，検索サービスは存在しなかった．ところが，その後のいわゆる Mosaic や Netscape の登場に起因するインターネットブームにより，インターネット上で提供される情報が急速に増加した．これをきっかけとして，WWW の情報を検索するための仕組みの構築が始まり，現在の AltaVista (`http://www.altavista.com/`) や HotBot (`http://www.hotbot.com/`) にれる検索サービスへと至っている．

WWW サーバ数は，1999 年初めに全世界で 400 万台を突破，2000 年 2 月には 1,000 万台のオーダーとなり，それらのサーバから発信される情報量は 17 億 URL を越えると推測される．さらに，現在もインターネットに接続されるコンピュータ台数の 2 倍以上の伸び率で WWW サーバは増え続けており，インターネット上で入手できる情報は，たった 1 年で 2 倍以上に膨れ上がっている．

しかし，検索サービスで検索できる URL 数は，FAST (`http://www.alltheweb.com/`)（公表されている収集 URL 数では最大を誇る検索サービスサイト）でも 3 億 URL 程度であり，全体の 1/5 以下である．検索できる URL 数が全 URL に占める割合（カバー率）は，年々減少する方向にあり，「欲しい情報」は検索できるかもしれないが「検索されて出てきた結果が全てではない」という状況が発生している．

検索サービスサイトは，大きく Yahoo! (`http://www.yahoo.com/`) のようなディレクトリ型と，AltaVista や FAST や HotBot のようなロボット型に分類できる．ディレクトリ型検索サービスでは，WWW のアドレスを示す URL(Universal Resource Locator) を，人手により，芸術，ビジネス，教育.., のように分野別に分類する方式をとっており，データ量が少ない反面，人手で索引や要約を作成するため，索引と要約の信頼度が高いといった特徴を持つ．一方，ロボット型検索サービスでは，WWW ロボットやスパイダーと呼ばれる Web 探索プログラムを用いて，インターネット上で見つけることのできる WWW サーバ上の情報を定期的に収集し，その情報の索引付けを行っており，情報量が多いという利点を持つ．逆に，各ページの要約を自動的に生成したり，索引付けを自動で行うため，要約の完成度が低いという欠点を持っている．

そして，ロボット型検索サービスを使っても，現在のインターネット上の急速な情報量増加に対応することができず，先に述べたように，検索できる URL 数が全 URL に占める割合が 1/5 以下になっている．

このような現状を打破するため，広域な範囲の Web ページをインターネット上に分散された複数の WWW ロボットで協調して高速収集するための「分散型 WWW ロボットの実験」を本研究で行った [2] [10] [16] [5] [11] [17]．以下では，これまでの研究開発及び実験結果を報告する．

2　Web ページ収集の現状と課題

本章では，WWW サーバ上のデータを高速に収集する手法の現状と今後の課題を明らかにする．特に，WWW(World Wide Web) はインターネットの一般市民への浸透に伴って急速に広まっており，WWW を通して世界中に分散するデータを巨大データベースとして活用できれば，その経済的効果は莫大なものになると推測される．

2.1　インターネットと WWW の発展

インターネットに接続するコンピュータ台数（ホスト数）は，米国 Internet Software Consortium (`http://www.isc.org/`) が半年毎に公表しているデータによれば，2000 年 1 月現在で約 7,200 万台である．また，英国 Netcraft 社 (`http://www.netcraft.co.uk/`) が毎月公表しているデータによれば，WWW サーバ数は 2000 年 1 月現在で 995 万台であり，全ホストの約 14 % が WWW サーバとして情報を発信していることになる．図 1に半年毎のインターネット接続ホスト台数とWWWサーバ数の推移を示す．また，インターネットに接続するホスト数，ドメイン数，WWWサーバ数の毎年の増加率（前年比）を図 2に示す．なお，ドメイン数は，edu, com, gov, mil, org, int, net で終わるドメインについては第二レベルで，それ以外のドメインは第三レベルで計算した．

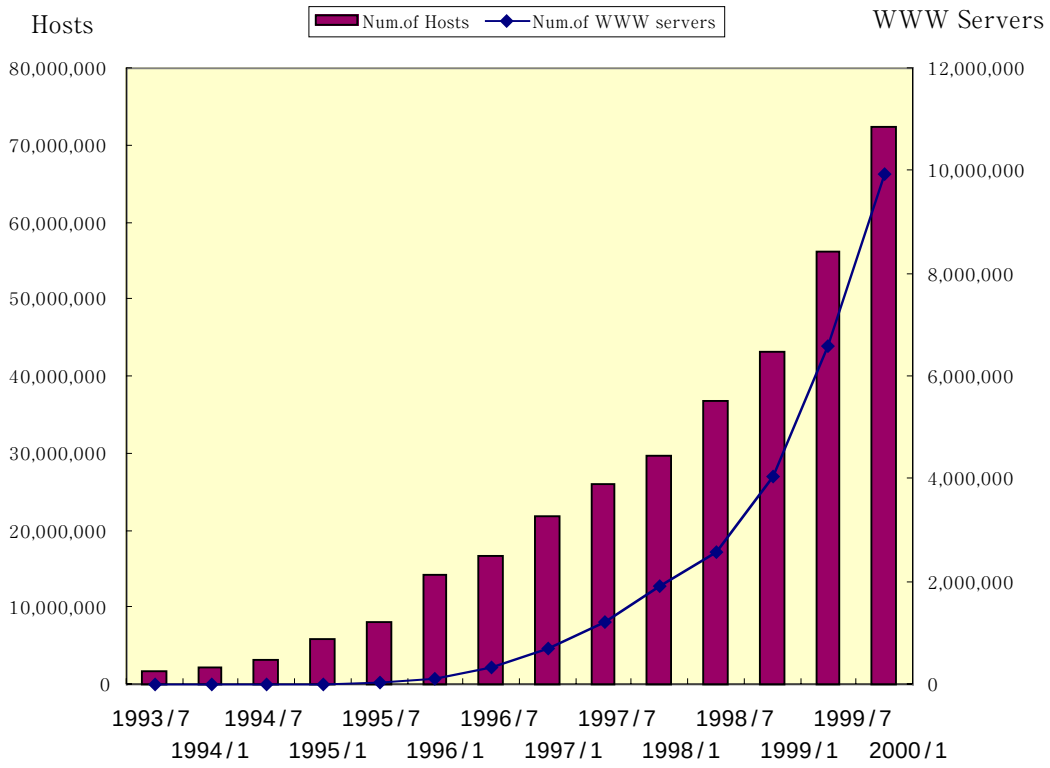


図 1: インターネットに接続するホスト数と WWW サーバ数の推移 (米国 Internet Software Consortium, 英国 Netcraft 社が公表するデータを元に作成)

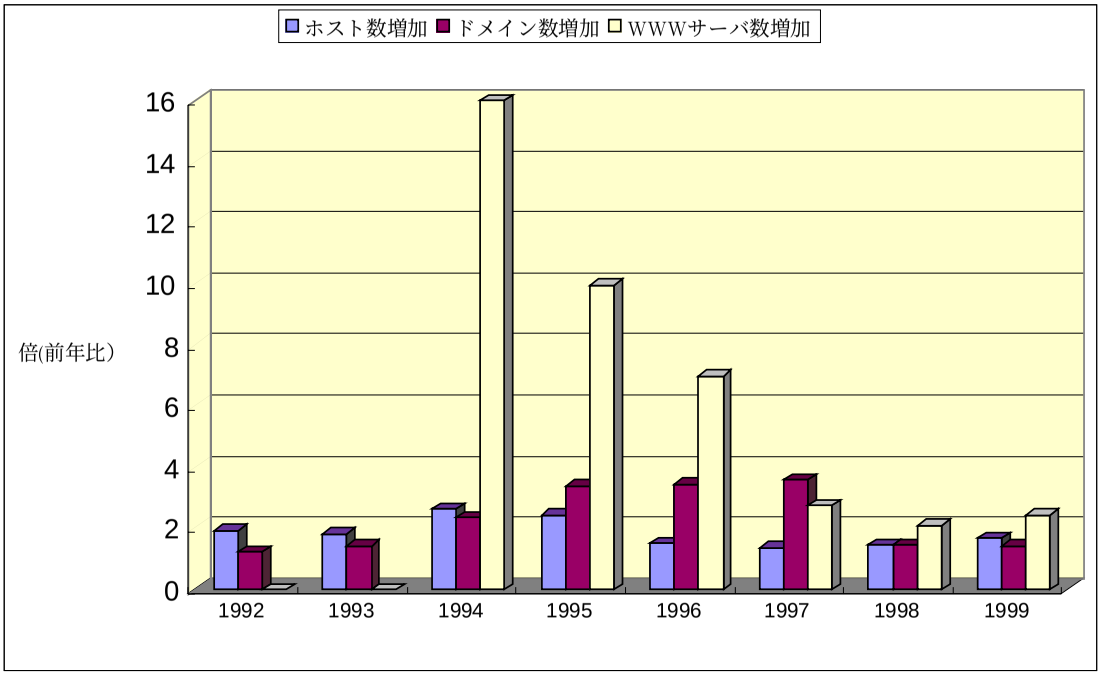


図 2: ホスト数, ドメイン数,WWW サーバ数の毎年の増加率 (米国 Internet Software Consortium, 英国 Netcraft 社が公表するデータを元に作成)

1994 年に WWW サーバ数が急増したのは，この年に Netscape のバージョン 1 の公開に伴って急激にユーザ層が広がり，それに伴ってデータを発信する WWW サーバ数が増加したためだと考えられる．その後，WWW サーバ数の増加率は減少したが，1998，1999 年とほぼ前年比 2 倍で推移している．一方，ホスト数及びドメイン数の増加は 1998，1999 年とほぼ前年比 1.5 倍で推移している．

2.2 WWW 情報検索サービスの動向と WWW サーバから発信されるデータ量

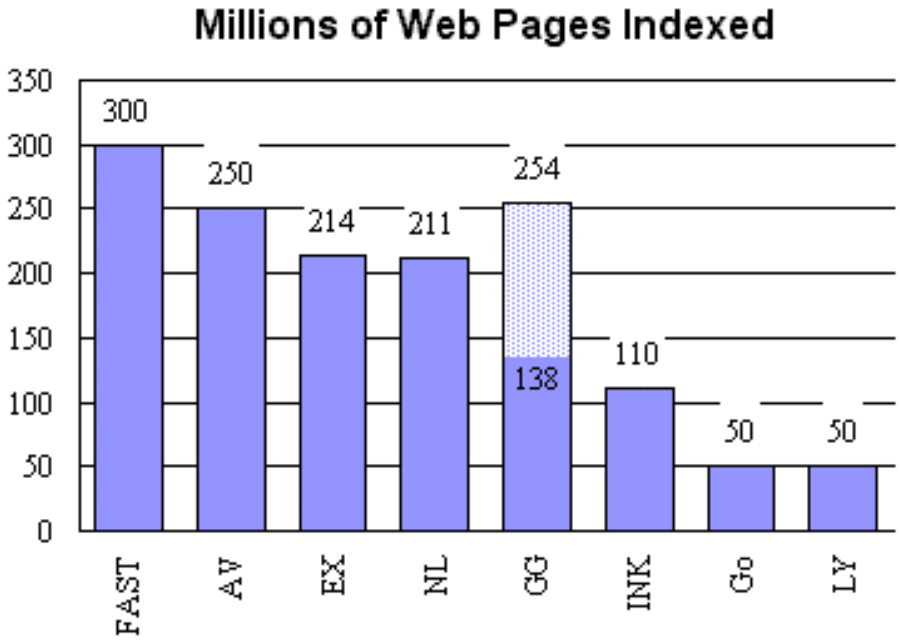
NEC 北米研究所の Steve Lawrence らの統計的調査によれば，1999 年 2 月現在の WWW サーバから発信されるデータ量は約 8 億ページと推測される [4]．テキストファイルのみのデータ量は 15TB (タグ部分を除けば 6TB) であり，平均 18.7 KB/page，メタタグを持つページが 1 ページ以上存在しているサーバ数は全体の 34.2 %，XML(Dublin Core) を用いているのは全体の 0.3 %であるとの結果が得られている．これらの推定の詳細は文献 [4] [3] に詳しいが，全体のページ数推定は，複数の検索サービスに対して同じ Query を送り，得られた検索結果の重複度から統計処理し算出している．また，各ページの平均の大きさ等は，実際に 2500 の WWW サーバのデータを収集し算出している．

1999 年 2 月時点での WWW サーバ数は，英国 Netcraft 社の調査によれば 430 万台であり，2000 年 1 月にはこれが 956 万台になっていることから推測すると，2000 年 1 月現在での Web ページ数は全世界で 17.8 億程度となる．

また，2000 年 1 月に開催された HICSS33 の国際会議における発表において，検索サービスサイトである Excite が発表したデータによれば，実験的に収集した

634,269,953 ページの内，英語で記述されているページが 453,685,690 ページ (71.5%)，日本語で記述されているページが 43,271,080(6.8%)，その他の言語で記述されているページが 21.7% である．

図 3に，2000 年 1 月現在での有名な WWW 情報検索サービスサイトが収集している Web ページ数（各社の公表値）を示す．これからわかるように，現在最も多くのページを収集している FAST(`http://www.alltheweb.com/`) でも，3 億ページ（全体の約 18 %）にとどまる．一方，1997 年末時点での Web ページ数は 3.2 億，この時点で最も多くのページを収集していたのは AltaVista の 1.2 億ページであり，全体の約 38 % をカバーしていた．このように，年々 Web ページ全体に対するカバー率は低下の一途をたどっている．



KEY: FAST=FAST, AV=AltaVista, EX=Excite, NL=Northern Light, GG=Google, INK=Inktomi, Go=Go (Infoseek), LY=Lycos.

original: <http://searchenginewatch.com/reports/sizes.html>

図 3: WWW 情報検索サービスが収集し検索対象としている Web ページ数

2.3 Web ページ収集と問題点

Web ページは，一般的には WWW ロボットと呼ばれるプログラムを使って収集する．収集の開始点となる URL を WWW ロボットに渡すことにより，WWW ロボットは，その開始点となる URL から http プロトコルを用いて順次リンクをたどり WWW ページを収集する．

公開されているデータ (`http://info.webcrawler.com/mak/projects/robots/active/html/`) によれば，2000 年 1 月現在での WWW ロボットの種類は 218 である．また，電子技術総合研究所 (`http://www.etl.go.jp/`) へのアクセスログを元に調査したところ，常に 20 を越える Web ロボットが WWW サーバのデータ収集を毎日行ってい

ることが判明した．さらに，電子技術総合研究所に対してアクセスのあるWWWロボットを調査(1999年7月12日～18日)したところ，全アクセスの37％がWebロボットによるものであった．詳細を表1に示す．

表1: 全アクセスに占めるWWWロボットのアクセスの割合(対象サーバ: 電総研)

WWWロボット運営サイト	全アクセスに占める割合（%）
Excite(`http://www.excite.com/`)	6 %
HotBot(`http://www.hotbot.com/`)	6 %
Infoseek(`http://www.infoseek.com/`)	4 %
FreshEye(`http://www.fresheye.com/`)	3 %

WIDEが集計した統計(`http://www.wide.ad.jp/`)によれば，httpプロトコルは，全プロトコルの70％程度を占めているため，電子技術総合研究所のWWWサーバへのロボットのアクセスが平均的なものであると仮定すると，「インターネットの約4分の1はWWWロボットが利用している」と推測することができる．

このように，

① 20を越えるWebロボットが同一のWWWサーバのデータを収集するのは無駄である点

② インターネットの約4分の1をWWWロボットが利用している点

③ WWW情報検索サービスがデータとして持つWebのページのカバー率が低下している点

を考え合わせると，複数のWWW情報検索サービスを提供するサイトが，個別にデータを収集せず協力して高速に集めるための仕組みの構築が重要なポイントとなる．

3　研究開発の目標と内容

本章では，本研究開発の目標とその内容について述べる．

3.1　研究開発の目標

WWW のデータを集めるサーチロボットが全ての WWW サーバにアクセスして必要な WWW データを収集するには，現状技術では，サーチロボットサーバーホストの性能および回線容量の制限などから，日本国内のみの WWW サーバを対象とした場合でも 1ヶ月以上の時間を要する．これに対して，WWW の世界ではデータの変化が非常に速く，これに追随するためにも，例えば日本全国の WWW データを時間のオーダーで収集するロボットの実用化が強く望まれている．

上記のような状況に対応するために，本研究は広域な範囲の WWW データを複数で協力して高速収集するための，分散協調サーチロボット（ソフトウェア）の研究開発を目的とし，24 時間以内に日本国内の WWW サーバのデータを収集することを目標とする．

3.2　ベースとして用いる理論・技術

以下の論文で提案された分散型 WWW ロボットの基礎技術を利用する．

田村健人, 分散 WWW ロボットに関する研究, 早稲田大学修士論文 (1997.3) [9]

3.3　前年度までの研究開発成果の年度毎の概略

平成１０年度の研究開発成果は以下の通りである．

早稲田大学 理工学部 村岡研究室，日本アイビーエム株式会社東京基礎研究所 ネットワーク＆ソリューション，京都大学大学院情報学研究科システム科学専攻 河野研究室，電子総合研究所 情報アーキテクチャ部の各研究者が共同で研究を実施した．

3.3.1　分散協調サーチロボットのプロトコルの検討

多数の WWW ロボットがお互いに協力して分散収集を行なうための，分担・通信プロトコルの設計を行った．また，プロトコルを実装するロボットを試作することにより，プロトコルとしての実用性を確認した．具体的には，分散された WWW ロボットを管理するためのサーバ PRSM (Public Robot Server Manager)，及び，PRS (Public Robot Server) の間のプロトコルを設計し報告書にまとめた．

3.3.2　分散エリア決定方法の検討

各 WWW ロボットの分担エリアを決定する方式について検討を行った．各 WWW ロボットの負荷を均等化するための「負荷均等化アルゴリズム」を提案し，分散を求めるためのプロトタイプシステムを作成した．そして，実データを用いて負荷均等化が実際に有効であることを確認した．

なお，当初の予定では，負荷均等化を行うために，ping コマンドによる時間を利用する予定であった．しかし，検討過程で，ping コマンドよりも，実際に WWW サーバにアクセスし一定のファイル (50 ファイル) を http により転送して転送速度を求める手法の方が正確であることが判明した．このため，転送時間の算出に，http プロトコルによる一定のファイル転送を行い，利用することとした．

3.3.3 分散協調サーチロボットプロトタイプシステムの設計と先行評価

分散協調サーチロボットプロトタイプシステムの設計を行った．さらに，収集したデータを配布する WWW ホームページデータ配布サービスの実現可能性についても評価を行った．

なお，当初の予定では，プロトタイプシステムを利用し，国内 50 個所程度の評価サイトに対してプロトタイプシステムの配備し，先行評価する予定であったが，「Java 処理系のバグによるプロトタイプ開発の遅延」及び「広域分散環境下でのデバッグ作業の困難性」から，7 サイトによる評価のみを行った．

3.4 今年度当初の研究開発計画概略

平成11年度の研究開発計画は以下の通りである.

3.4.1 分散協調サーチロボットの開発

平成１０年度に研究開発を行なったサーチロボットのプロトタイプに対して，以下の機能追加および改良を行なう.

1. **負荷分散単位細分化アルゴリズムの検討及び実装**

   試作時の負荷分散単位はWWWサーバ単位であった．しかし，特定のWWWサーバは数万URLのデータを持つものがある．このような大規模サーバに対しては，複数のPRSで協調させることにより，収集時間の短縮を図ることが必要不可欠である．このための仕組みを検討し，分散協調サーチロボットに実装する.

2. **負荷変動対応アルゴリズムの検討及び実装**

   WWWサーバは，曜日さらには時間帯によって負荷変動が大きい．このため，高速収集を可能としつつ，WWWサーバへ与える負荷を小さくするためには，WWWサーバの能力や負荷によって，収集間隔(現在は20秒毎に1URL取得)を動的に変更する仕組みが必要不可欠である．このための仕組みを検討し，分散協調サーチロボットに実装する.

3. **分散エリア決定アルゴリズムのサーチロボットへの実装**

   平成１０年度に研究開発を行った「分散エリア決定方法」を分散協調サーチロボットに実装し，新規にWWWサーバが発見された場合，人手を介さずに自動的に割り当てることができるようにする.

3.4.2 分散協調サーチロボット環境の実装

上記で開発した分散協調サーチロボット，収集データ再配布環境，及び，分散協調サーチロボットを動作させるための環境であるSolaris，Java等を日本全国の30以上のサイトに実装する.

1. **システムの自動メンテナンス方式の開発・実装**

   広域に分散した複数のコンピュータを使った実験を行う際には，システムのメンテナンス性を高めることが実験効率化の必須条件となる．このため，分散

協調サーチロボットのソフトウェアを自動的にバージョンアップする仕組みを実装する.

2. **収集データ再配布環境の構築**

   各分散協調サーチロボット毎に収集したWWWデータを検索サービスサイトに配布するための仕組みを構築し，実装・サービス提供を行なう.

3. **サーバのフォルトトレランス性確保の方式設計**

   ＰＲＳＭ又は一部のＰＲＳが故障した場合の代行の仕組みについて方式設計する.

3.4.3 分散協調サーチロボットの評価

各実装後，以下の各々の項目について本システムを評価する．評価にあたっては，日本全国に配置された30以上の分散協調サーチロボットを用いる.

1. 負荷均等化評価　平成１０年度に開発した「分散エリア決定方法」，及び，上記の技術を用いることにより，各分散協調サーチロボットの負荷が均等化されているかどうかについて評価する.
2. 分散協調サーチロボット総合評価　開発した分散協調サーチロボットを用いて日本国内の全WWWサーバを対象とした収集を開始することにより，本システムにより得られる性能向上（収集時間短縮効果）について評価する.

3.5 研究開発内容概略

本節では，本実験への参加組織，研究開発の概要について述べる．

3.5.1 実験参加者

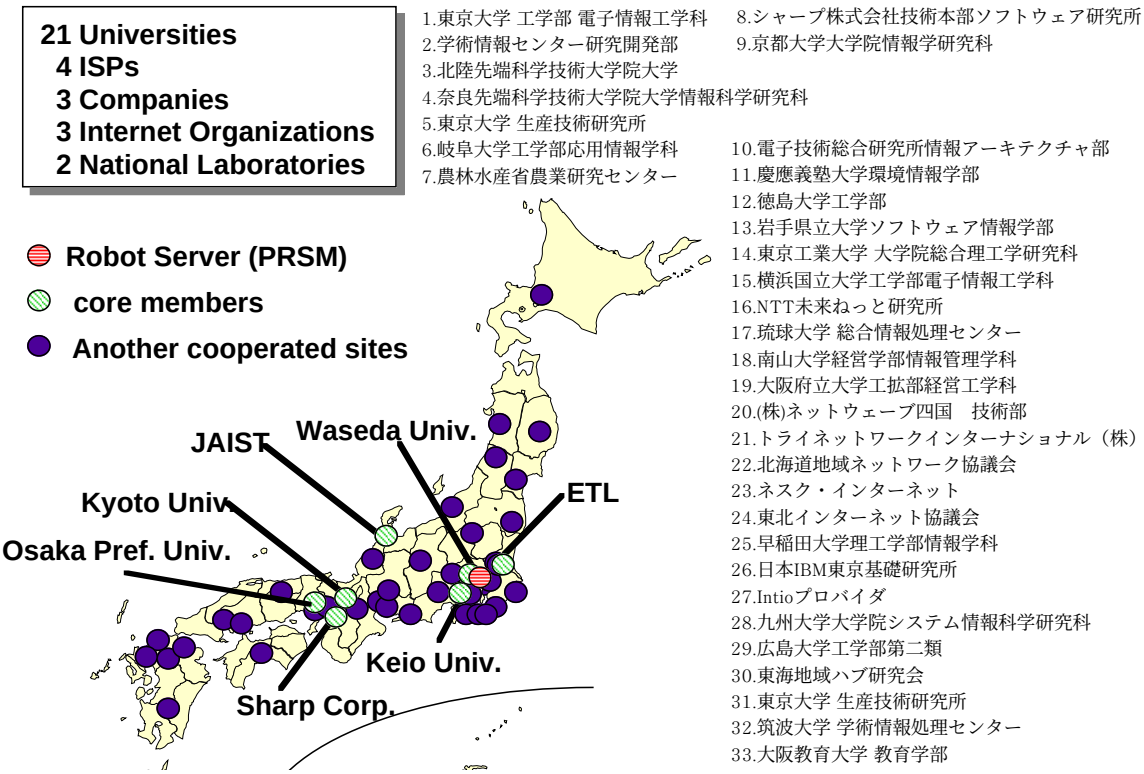


図 4: 実験参加組織

コアメンバとして，早稲田大学，京都大学，北陸先端大学院大学，慶應義塾大学，大阪府立大学，日本IBM(株) 東京基礎研究所，シャープ (株)，電子技術総合研究所の 8 研究機関が参加すると共に，外部協力機関を併せて合計 33 機関（21 大学，4 インターネットサービスプロバイダ，3 企業，3 ネットワーク機関，2 国立研究所）が参加している (図 4)．

3.5.2 分散型 WWW ロボットの概要

分散型 WWW ロボットでは，

① WWW ロボットをネットワーク上に分散して複数配置する．

② 分散したWWW ロボットが担当するサーバを自動的に決定させると共に，協調動作させる．

の 2 点により，高速にWWW サーバ上のデータを収集することを目指しており，直接の効果・成果，及び，波及効果として以下の 4 点が期待される．

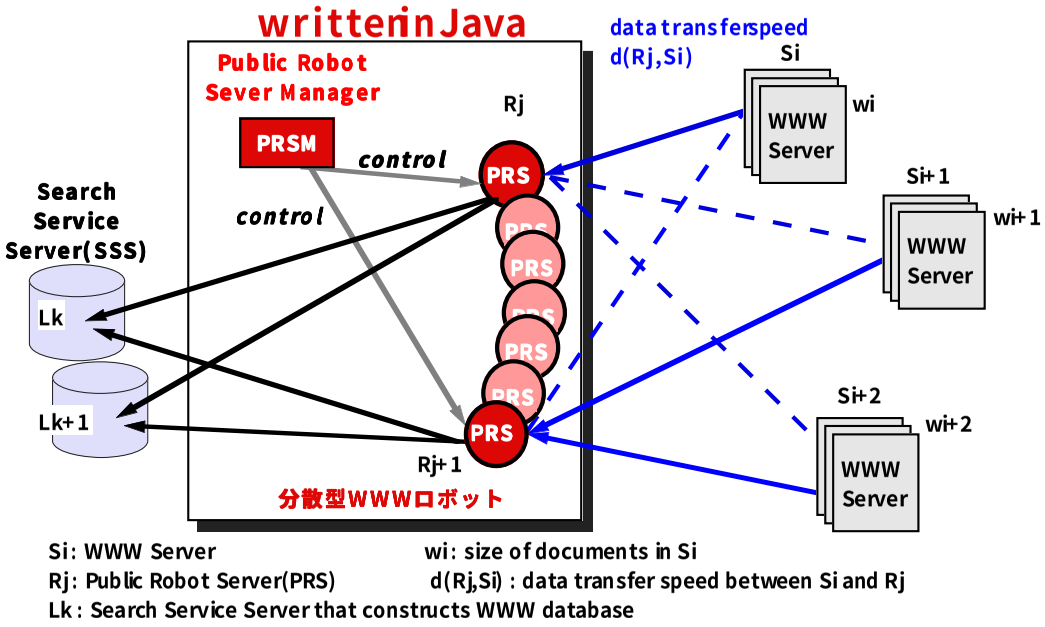


図 5: 分散型 WWW ロボットの仕組み

① WWW サーバ上のデータを高速収集（目標は日本全国の WWW サーバ上のデータを 24 時間以内に収集) することが可能となる.

② 高速収集により，検索サービスサイトでは，最新のデータを利用した検索サービスの提供が可能となる．つまり，最新性において質の高い検索を提供することができるようになり，インターネット検索ビジネスの活発化にもつながると予想される.

③ 現在各検索サービスサイト毎に独立に動かしている WWW ロボットを本研究開発によって開発された広域分散協調サーチロボットに置き換えることにより，国レベル，あるいは，世界レベルでの協調収集が可能となる．これにより，現在インターネットの負荷の４分の１を占める WWW ロボットによる負荷を大幅に削減することが可能となる.

④ 本ソフトウェアは次世代のコンピューティング環境として近年研究開発が活発化の兆しを見せている Globus [1] に代表される「広域分散コンピューティング」の一つの重要なアプリケーションとなり，本分野の研究開発の活性化に寄与する.

3.5.3 分散型 WWW ロボットの動作概要

分散型 WWW ロボットは，図5に示すように，全体を管理する Public Robot Server Manager(PRSM) と個々の WWW ロボットである Public Robot Server(PRS) から構成される.

PRSM は，PRS に対して担当 WWW サーバの分配や，各 WWW サーバと PRS 間との距離計測を指示する．一方，PRS で新規に発見された WWW サーバや距離計

測の結果は，PRSM に送られ，PRSM 側で PRS への担当 WWW サーバ割り当てを行う．PRS への担当 WWW サーバ割り当て方式として，ランダム方式と負荷均等化方式の二種類をサポートしている [13]．このようにして，PRS は PRSM からの指示に基づき各々互いに重複しない WWW サーバを担当し Web ページを収集する．

収集されたデータは，最終的に図中の Search Service Server(SSS) に再配布することにより，検索サービスのための索引作成を行う．現在の HTTP/1.1 準拠のサーバは，1 回の接続で複数のデータを転送する Keep-Alive 機能を持っているが，複数のデータをまとめて転送する機能は持っていない．このため，各 PRS で収集したデータを SSS に再配布する際のオーバヘッドを小さくするため，複数のデータを 1 回の接続でまとめて圧縮して送る機能，及び，SSS からの要求に応じてデータ中の必要な部分のみ (例:URL や更新日時の指定による指定) を送る機能を PRS に持たせる．

現在，PRS・PRSM は Java1.2 で実装されており，PRSM は WWW サーバ Apache に JServ モジュールを組み込み，処理は Java servlets で行っている．

PRS は，まず起動時に PRSM にアクセスし，担当リストを PRSM から取得する．担当のサーバのページをすべて取得し終ると，再び担当リストを取得する．

ページ収集は，インターネット及び相手の WWW サーバへ与える負荷を最小限におさえるため，動作時間を制限し，**平日は午前 2 時〜午前 8 時，土日は午前 2 時〜午後 10 時**のみページ収集を行う．これは，Internet Exchange サイトの調査で，午前 2 時〜午前 8 時頃の負荷が一番小さいとの結果に基づいている．さらに，Java のスレッドを用い，同時に 200 個のサーバに対して並列にアクセスを行ない，高速化を実現している．

1 つのサーバに対しては，サーバへの負荷を少なくするため，連続してアクセスはせず 20 秒おきにアクセスを行なうが，WWW サーバが HTTP/1.1 の Keep Alive 機能に対応している場合は可能な限り連続でアクセスを行なう仕様となっている．

4　今年度の活動状況

学校法人早稲田大学の指導のもとに，33 組織 (図 4) が共同で研究を実施した.

4.1　分散協調サーチロボットの開発

当初の計画通り，平成 10 年度に研究開発を行なったサーチロボットのプロトタイプに対して以下の機能追加および改良を行った．分散協調サーチロボットの仕様・利用方法については，以下の 5にまとめた.

4.1.1　負荷分散単位細分化アルゴリズムの検討及び実装

負荷分散単位として新たに「各 Web サーバのディレクトリ単位」での分散を可能とし，プロトタイプシステムへ実装した.

詳細は，5.1.5に示す.

4.1.2　負荷変動対応アルゴリズムの検討及び実装

高速収集を可能としつつ，WWW サーバへ与える負荷を小さくするために，PRS と WWW サーバとの間の通信速度に応じてアクセス間隔を変更する仕組みを構築し実装した．具体的には，PRS と Web サーバとの HTTP プロトコルを用いた通信において「（総転送量÷所要時間）の値が，その PRS が担当する WWW サーバ内の上位 20% 以上」である場合に，アクセス間隔を標準の 20 秒から 10 秒に短縮する.

詳細は，5.1.4に示す.

4.1.3　分散エリア決定アルゴリズムのサーチロボットへの実装

「分散エリア決定方法」を分散協調サーチロボットと同一のマシン上に実装し，新規に WWW サーバが発見された場合の自動割当てを実現した.

詳細は，5.2.2に示す.

4.2　分散協調サーチロボット環境の実装

当初の計画通り，上記で開発した分散協調サーチロボット，収集データ再配布環境，及び，分散協調サーチロボットを動作させるための環境である Solaris，Java 等を日本全国の 33 のサイトに実装した.

4.2.1　システムの自動メンテナンス方式の開発・実装

システムのメンテナンス性を高めるため，分散協調サーチロボットの PRS を自動的にバージョンアップする仕組みを実装した.

本システムの仕様・利用方法を 5.3に示す.

4.2.2 収集データ再配布環境の構築

各分散協調サーチロボット毎に収集したWWW データを検索サービスサイトに配布するための仕組みを構築し，実装・サービス提供を行える環境を整備した．具体的には，各PRS が収集したWWW データを「データ収集システム・データ配布システム」と呼ばれるコンピュータに集中して集め，あらかじめ登録されている「第三者クライアント」にのみWWW データを配布できる仕組みを構築した．

本システムの仕様・利用方法を5.4に示す．

4.2.3 サーバのフォルトトレランス性確保の方式設計

PRSM 又は一部のPRS が故障した場合についての代行方法について方式設計を行った．PRSM に関しては二重化を行うと共に，PRS については故障したPRS が担当するWWW サーバを自動的に他のPRS に振り替える手法を設計した．

4.3 分散協調サーチロボットの評価

当初の予定通り，以下の項目について本システムを評価した．なお，当初の計画では，30 以上の分散協調サーチロボットにより評価を行うことになっていたが，以下の理由により，評価は17 個の分散サーチロボットを用いて実施した．

a. 本プログラムのプラットフォームとなるJava のスレッド処理のバグが改善されたJava2(v.1.2.1_03) がリリースされたのが2000 年1 月であり，この時点から正式な実験を開始した．このため，当初予定していた実験期間が大幅に短縮された．

b. 一回の実験に約2 週間を要し，この間中30 以上のPRS 全てが100 %動作していることを保証して実験を行うことが困難であった．このため，実験は全サイトで行うが，評価結果としては，これらのPRS で共通して集めることのできたデータのみを用いて比較した．

4.3.1 負荷均等化評価

日本国内に分散された17 個のPRS を用いて，国内のWeb サーバ6,500 台のデータ収集を行った．この結果，ランダムに分散した場合に比較し，負荷均等化によって11.1 倍の高速化が図られた．また，ランダム分散では，実行時間の一番短いPRS と一番長いPRS との差が37.1 倍であったが，負荷均等化によって，9.9 倍まで小さくすることができた．このことから，各分散協調サーチロボット(PRS) の負荷が均等化されていることが確認できた．

詳細な評価結果を6.3に示す．

4.3.2 分散協調サーチロボット総合評価

単一のPRS で6,500 台のWeb サーバを収集する時間は，17 台のPRS によって異なるが，各PRS が逐次に収集した場合で最小28 日～最大712 日，各PRS が200 スレッ

ドで並列に収集した場合でも最小23時間(NTT未来ねっと研究所)～最大23日(北海道地域ネットワーク協議会)が必要であった．これを負荷均等化分散によって，高速化することにより，17台のPRSで6.3倍(NTT未来ネット研究所を基準)～286倍(北海道地域ネットワーク協議会を基準)の高速化を達成することができた．

詳細な評価結果を6.4に示す．

5　分散協調プロトコル仕様及び利用方法

本章では，研究開発を行った「分散型ロボット(PRS/PRSM)」「分散エリア決定アルゴリズム」「収集データ再配布方式」「プログラム自動バージョンアップ方式」についてのプロトコル仕様及び利用方法を詳細に説明する．

5.1 PRSM/PRS

本節では，分散型WWW ロボットのプロトコル設計と利用方法について述べる．

5.1.1 プロトコル設計

分散型WWW ロボットである PRS (Public Robot Server) と，PRS 群を制御する PRSM (Public Robot Server Manager) との間のプロトコルを以下の５つのメッセージに分類し設計した．

５つのメッセージは，以下の通りである．

(M1) 担当するサーバのリスト配布メッセージ (PRSM → PRS)

(M2) 発見したサーバのリスト報告メッセージ (PRS → PRSM)

(M3) 収集ログファイル転送メッセージ (PRS → PRSM)

(M4) リスタート要求メッセージ (PRSM → PRS)

(M5) 参加・脱退メッセージ (PRS → PRSM)

本システムにおける PRS と PRSM の動きは，以下の通りである．

① PRSM は各 PRS に各々の担当サーバリスト (M1) を送信

② 各 PRS は担当サーバリストに従い WWW ページを収集

③ 各 PRS は収集 WWW ページを解析し未知の WWW サーバを PRSM に通知 (M2)

④ 各 PRS は WWW ページ収集情報 (サイズ, 転送時間) を PRSM に通知 (M3)

⑤ PRSM は各 PRS から送られた情報を元に，次の担当サーバリストを作成

PRS がファイアウォール内に設置される場合も考慮し，PRS・PRSM 間の通信はすべて HTTP 上で行なう．さらに，PRS からの PRSM に対するポーリング方式を採用することにより，PRS と PRSM との間の非同期性を確保した．(M4) は，評価実験を効率的に行うためにに付加されたメッセージであり，PRSM 側から強制的に PRS の担当サーバを変更することができる．

5.1.2 PRS・PRSM 間メッセージの詳細

以下の説明で，*PRSM* はPRSM のホスト名，*PRSNAME* はアクセスするPRS のホスト名である．

(M1) 担当するサーバのリスト配布メッセージ (PRSM → PRS)

`http://`*PRSM*`/servlets/list?host=`*PRSNAME*

PRS はPRSM の特定のURL にアクセスし，返答として担当サーバのリストを取得する．担当サーバリストは，URL もしくはURL とそのURL のリンク元URL を1 行に書いたもの (例: http://www.etl.go.jp:80/) で，それを gzip で圧縮して転送する．以下にデータ形式を示す．

```
     ServerList  ::=  gzip(ServerListItem*)
 ServerListItem  ::=  URL (HT RefererURL)? LF
                      | MaxTransfer
                      | Comment
    MaxTransfer  ::=  '#MaxTransfer=' [0-9]+ LF
            URL  ::=  <URL; RFC2396>
     RefererURL  ::=  <URL; RFC2396>
        Comment  ::=  '#'[^LF]* LF
             HT  ::=  <US-ASCII HT, horizontal-tab (9)>
             LF  ::=  <US-ASCII LF, linefeed (10)>
```

WWW サーバを各PRS に分担させるときに，各PRS において各WWW サーバのアクセス速度 (PRS-WWW サーバ間のデータ転送速度) を計測する必要がある．計測に関しては各PRS が全WWW サーバに対しアクセスする必要があり，このときはWWW サーバの全ページを収集する必要はない．上記ルールにおけるMaxTransfer は，各WWW サーバのうちいくつのページをアクセスするかを指定する．現在の実装では，最初に'MaxTransfer=50' を指定し各WWW サーバの50 ページを収集し，そのログをPRSM で集計してから本当の分担を決定する．

(M2) 発見したサーバのリスト報告プロトコル (PRS → PRSM)

`http://`*PRSM*`/servlets/record`

PRS はPRSM の特定のURL にアクセスし，HTTP のPOST メソッドにより発見サーバリストを送信する．形式はMaxTransfer 指定に意味がない意外は担当サーバリスト (前記M1) とまったく同一である．gzip で圧縮する点も同じである．

```
NewServerList  ::=  gzip(ServerListItem*)
```

(M3) 収集ログファイル転送メッセージ (PRS → PRSM)

`http://`*PRSM*`/servlets/submitlog`

PRS は PRSM の特定の URL にアクセスし，HTTP の POST メソッドにより収集ログを送信する．形式は第 1 行目 (最初の LF まで) がファイル名で，それ以降はログを gzip 圧縮したものである．

```
      Log  ::=  FileName LF gzip( (LogLine | Comment)* )
 FileName  ::=  PRSNAME '.' 日付
  LogLine  ::=  '"' URL '",' HTTP 要求サイズ ',' HTTP 応答サイズ ',
                ' 所要時間 ',"' HTTP 応答ヘッダの中身 '"' LF
```

(M4) リスタート要求メッセージ (PRSM → PRS)

実験の便のための「リスタート要求」のメッセージである．PRS は 1 時間毎に `http://`*PRSM*`/servlets/isrestart?host=`*PRSNAME* に GET メソッドでアクセスし，その返答の 1 行目が "`RESTART`" の場合，現在の担当サーバリストの破棄・ディスク上にある収集したページの削除を行ない，あらたに担当サーバリストを取得する．

(M5) 参加・脱退メッセージ (PRS → PRSM)

`http://`*PRSM*`/servlets/subscribe?host=`*PRSNAME*

`http://`*PRSM*`/servlets/unsubscribe?host=`*PRSNAME*

PRS が参加するとき・脱退するときは上記のような URL に HTTP の GET メソッドでアクセスする．

5.1.3 PRS・PRSM のプログラム仕様

PRS・PRSM の主要部分を Java で実装した．PRS は完全な Java アプリケーションである．PRSM は WWW サーバ Apache に JServ モジュールを組み込み，処理は Java servlets で行う．

なお，2000 年 2 月末現在，JDK のバグ[1]により，*JDK*1.2.1_03 SDK(Solaris Production Release) 以降の JDK [2]でのみ動作することを確認している．また，動作確認済みのプラットフォームは，Solaris8/Sparc, Solaris2.6/x86, Solaris2.6/x86 である．

[1] JDK1.2 には，同期回りの実装ミスでデッドロックするというバグが存在する．このバグは JDK 1.3 で解決される予定．

[2] Reference Implementation では動作しない．これは，Reference Implementation は，largefile aware でないいう本質的な問題もあり，Production Release の JIT の生成したコードが原因でプログラムが落ちるという致命的な問題があるため．

5.1.4 PRSの動作

PRSは，まず起動時にPRSMにアクセスし，担当リストをPRSMから取得する．担当のサーバのページをすべて取得し終ると，再び担当リストを取得する．

ページ収集は時間を制限している．平日は午前2時～午前8時，土日は午前2時～午後10時のみページ収集を行う．

一日に一度，午前9時ごろ[3]，収集ログをPRSMに転送する．

初期設定では同時に200個のサーバに対してアクセスを行なう．1つのサーバに対しては20秒おきにアクセスを行なう．ただし，WWWサーバがHTTP/1.1のKeep Alive機能に対応している場合は可能な限り連続でアクセスを行なう．

WWWサーバの反応が速い場合には，アクセス間隔を20秒より短くする．PRSが「このWWWサーバは速い」と判断する基準は，PRSとWWWサーバとのHTTPプロトコルを用いた通信において，

(総転送量÷所要時間)

の値が，その担当するWWWサーバ内の上位20%以上かどうか[4]，である．すなわち，PRSが担当するWWWサーバ内で，そのWWWサーバとの間の通信速度が早い1/5のWWWサーバに対しては，WWWサーバの能力にまだ余裕があると判断する．

WWWサーバの能力にまだ余裕があると判断した場合，転送間隔をデフォルトの20秒から10秒に短縮する．

5.1.5 インプリメント上の注意点

上記のプロトコル仕様に基づき，PRSとPRSMを試作した．インプリメント上で，いくつかの動作上の制限があるので，以下に詳細を示す．

- 仕様では，PRSからPRSMへ収集ログを転送した後，PRSMが集計を行ない新規発見WWWサーバ等を含めて担当サーバリストの更新を行なう．本来は，PRSMの機能として実装すべきであるが，現在は，別プログラムとして動作し，スクリプトにより連動する仕組みとなっている．

- プロトコル上は，担当サーバリスト・発見サーバリストはサーバのトップページだけでなく任意のページを入れることができる．この機能を使って，ディレクトリ単位での負荷分散も可能となっている．

[3] PRSは起動後20分毎に現時間をチェックしており，午前9時を過ぎた時点(従って午前9時～午前9時20分の間)で本処理を実行する．

[4] 実験データからデータ転送速度14.0byte/ms以上が上位20%に相当することがわかったため，14.0byter/ms以上の転送速度になった場合に10秒に短縮する．

- プロトコル上は，referer 情報 (そのページを発見することになったリンク元の URL) を含めることが可能である．しかし，情報交換によるトラフィック増加を考え，実装では「サーバのトップページのみ・referer なし」でやりとりしている．これにより，現在の PRS は，あるサーバのトップページからそのサーバ内のリンクだけで辿れないページを収集することができない．トップページだけでなく，すべてのページの情報を送受信した場合にどの程度のトラフィックが発生するかを検証する必要がある．

- PRS の動作環境は，今回はほぼすべて Solaris 2.6 Intel Edition であるが，このプラットフォーム用の Java Virtual Machine (JVM) で PRS のプログラムを動作させると非常に不安定である (ハングアップ，コアダンプなど)．この問題を回避するため，PRS のプログラムは毎日 0 時ごろに起動し，朝 9 時 (土日は夜 11 時) にプロセスを終了するように設計した．ハングアップなどでプロセスが動き続けているときは，0 時の起動時にプロセスを強制的に終了させる．

5.1.6 PRSM プログラムの利用方法

インストール

1. Java2 v1.2[5] が動作する Servlet 動作環境を用意する.

2. 作業用ディレクトリとプログラムディレクトリを作成しておく. 作業用ディレクトリは, 動作中に必要なデータなどを読み書きする場所なので, 大きな容量が必要である. プログラムディレクトリは, 配布パッケージが展開できればよい. 以下の説明で, 作業用ディレクトリを'/data/prsm', とする.

3. 配布パッケージ prs-2000mmddn.zip もしくは prs-2000mmddn.tar.gz を, プログラムディレクトリで展開する.

4. プログラムディレクトリの org/hauN/kent/prsm/prsm.properties の'/data/prsm/' の部分を作業用ディレクトリに変更する.

5. Servlet 動作環境において, パッケージを展開したプログラムディレクトリが CLASSPATH に含まれるように設定する.

6. Servlet 動作環境において, 以下のクラスが指定の名前の Servlet として起動できるように設定する.

クラス名	Servlet 名
org.hauN.kent.prsm.IsRestart	/servlets/isrestart
org.hauN.kent.prsm.GetList	/servlets/list
org.hauN.kent.prsm.ListRobot	/servlets/listrobot
org.hauN.kent.prsm.RegisterServer	/servlets/record
org.hauN.kent.prsm.Restart	/servlets/restart
org.hauN.kent.prsm.RegisterLog	/servlets/submitlog
org.hauN.kent.prsm.Subscribe	/servlets/subscribe
org.hauN.kent.prsm.Unsubscribe	/servlets/unsubscribe

運用

1. 担当サーバの設定
   作業ディレクトリ下の'assign/' ディレクトリに, PRS の名前で担当サーバリストのファイルを置く. 登録されている PRS の名前は, 作業ディレクトリの robothost.list ファイルに羅列されている. 担当サーバリストは, プロトコルの (M1) そのものである. ただし, 'http://' で始まらない行があると, (M1) の送信時に自動的に'http://' を補完する.

2. ログ
   作業ディレクトリ下の'logs/' ディレクトリに, (M3) で送られた各 PRS の転送ログが格納される. ファイル名は, ＜PRS 名＞.yyyymmddhhMM である (yyyymmddhhMM は転送された日付と時刻). PRSM の動作のログが作業ディレクト

[5] JDK1.2.1_03 以降

リの'prsm.log' に記録される．このファイルは問題が発生したときの原因究明用に存在するので，必要なければ削除して構わない．

3. PRS の監視と制御
'http://PRSM のマシン /servlets/listrobot' にアクセスすると，PRSM が把握している各 PRS の状況を表示する．具体的には，各 PRS について，

・リスタート要求を処理したかどうか
・最後にリスタートチェックをした時刻
・最後に担当リストを送った時刻

を表示する．
PRS が動いていれば PRS から PRSM にリスタートチェックが 1 時間に 1 回あるはずなので，リスタートチェックが長い時間なければ，この PRS は動作していないことがわかる．'http://PRSM のマシン /servlets/restart' では，

・次に送る担当サーバリストが 50 ページ限定なのかページ数無制限なのか
・各 PRS をリスタートさせるかどうか

を設定することができる．

5.1.7 PRS プログラムの利用方法

インストール

1. Java2 v1.2[6] が動作する環境を用意する．
2. 作業用ディレクトリとプログラムディレクトリを作成しておく．作業用ディレクトリは，動作中に必要なデータなどを読み書きする場所なので，大きな容量が必要である．プログラムディレクトリは，配布パッケージが展開できればよい．以下の説明で，作業用ディレクトリを'/data/prs'，プログラムディレクトリを'/data/class' とする．
3. 配布パッケージ prs-2000mmddn.zip もしくは prs-2000mmddn.tar.gz を，プログラムディレクトリで展開する．
'/data/class' 以外のディレクトリに展開する場合は，'robot.conf' および'prsinit.sh' を書き換える必要がある．

```
% mkdir -p /data/class
% cd /data/class
% /path/to/gzip -dc /path/to/prs-xxxx.tar.gz | tar xvf -
```

[6] *JDK*1.2.1_03 以降

4. プログラムディレクトリにある'prs.conf' のファイル名を変更し，編集する．ファイル名を変更するのは，プログラムのバージョンアップの際に上書きされるのを防ぐためである．以下の説明では'prs.conf' を'robot.conf' に変更したとする．

   変更を最小に抑えるには，必ず'/data/class' で配布パッケージを展開し，作業ディレクトリを'/data/prs' にする．

   % cp prs.conf robot.conf
   % mkdir -p /data/prs/html
   % vi robot.conf

   して，prs.myfqdn (と，必要なら prs.proxy) を設定
   以下に，'robot.conf' の各項目の意味を説明する．

項目名	説明
prs.myfqdn	ロボットプログラムを動かすホストのホスト名(ドメイン名込み)． DNS の名前を持っていないときに限り，IP アドレスでも構わない．
prs.logfilename	ロボットプログラムの動作中のログを記録するファイル名．
prs.transferlogfilename	ロボットプログラムの動作中の転送記録を保持するファイル名．
prs.serverlistfilename	担当情報を記録しておくファイルのファイル名．
prs.datadirectory	ロボットで集めてきたファイルを格納するディレクトリ名． 大容量が必要．
prs.proxy	ロボットを動かすホストがファイアウォールの中にある場合や， キャッシュを通したい場合などに proxy.foo.co.jp:8080 のようにして proxy を指定する． proxy がいらない場合は，「prs.proxy=」とする．

   他に，PRSM の各用途用の URL が記述されている．PRSM が既定値と異なる場合は，各 URL も変更する．

5. crontab を編集する．
   % crontab -e
   10 0 * * * /data/class/prsinit.sh start 2 > /dev/null
   と入力してエディタを終了する．

6. ロボットプログラムの準備ができたことを PRSM に通知する．
   % /usr/java1.2/bin/java -cp /data/class RobotSubscribe /data/class/robot.conf

運用

1. **ホストの電源を落としたいとき**

   ps コマンドで java のプロセスが動いていなかったら，そのまま shutdown して構わない．java のプロセスが動いているときは，プログラムディレクトリ

(’/data/class/’) の prsinit.sh を
“% /data/class/prsinit.sh stop”
のように実行してから shutdown する．

なお，ロボットは，毎日 cron により 0 時 10 分に起動される．また，平日は，午前 8 時まで収集，土日は午後 10 時まで収集し，その後ログが転送される．このため，平日は午前 11 時半～午前 0 時，土日は，午後 11 時半～午前 0 時であれば，java のプロセスは動作していないため，収集に影響を与えずにホストの電源を落すことができる．

2. **ロボットプログラムのバージョンアップ**

上記の方法でプログラムを停止，新しい配布パッケージを，以前展開したディレクトリと同じところに展開して上書きする．また，自動化するには，本章で説明する自動バージョンアッププログラムを用いる．

3. **ログファイル**

作業ディレクトリにある ‘robot.log’ は，PRS 動作の状況を記録したものである．基本的に不必要であるので，ファイルが邪魔になったら削除しても構わない．‘transfer.log’ は転送の状態を記録したファイルで，これは定期的に PRSM に送られる．PRSM に送られると ‘transfer.log.yyyymmddhhMM’ という名前に変更されるので，名前が変更された後は削除して構わない．ただし，ネットワークの問題等により PRSM への転送が正常に終了しない場合のために 1 週間程度のログを残すのが望ましい．

5.2 分散エリア決定

本節では，PRS が担当する WWW サーバを決定する手法について述べる．

5.2.1 アルゴリズム

コストの定義

各 PRS は WWW サーバ（以下サーバと記す）を単位とした Web ページの収集を行い，PRS 間で担当サーバに重複はないものとする．各 PRS の収集コストは，担当サーバ内の全ての Web ページを収集するのに要した時間で次のように定義する．

PRS $P_i(i = 1, 2, ..., m)$ の担当サーバ $S_i = \{s_{i1}, s_{i2}, ..., s_{in_i}\}$ を収集するのに要した時間をそれぞれ $C_{ij}(j = 1, 2, ..., n_i)$（単位は msec）とすると，PRS$P_i$ に対するコスト $Cost(P_i)$ は，次式で与えられる．

$$Cost(P_i) = \sum_{j=1}^{n_i} C_{ij} \quad (i = 1, ..., m)$$

担当サーバに重複はないので，以下の条件が成立する．

$$S = \bigcup_{i=1}^{m} S_i,\ \forall(i, j), i \neq j, S_i \bigcap S_j = \phi$$

また，全 PRS の収集が完了したときに，収集システムの収集が完了したと考える．したがって，収集システムのコストは，各 PRS のコストの最大値

$$\max\ Cost(P_i) \tag{1}$$

で与えられる．提案するアルゴリズムの目標は，(1) 式の最小化を目指すこと，すなわち

$$\min\{\max Cost(P_i)\} \tag{2}$$

である．

PRS 間の情報交換

PRS 間で互いに情報交換を行いながらコストを均等化するプロセスを実行するアルゴリズムを提案する．ただし，PRS の数が増えるとメッセージ量の増大による問題が生じる．したがって，PRS 間の情報交換に制約を与え，距離が近い同士の局所的な情報交換，すなわちネットワークルーティングと同様に各 PRS 間のネットワーク距離を考慮する．そこで，プリムのアルゴリズム [7] を用いて図 6のような最小木を与え，最小木に沿う隣接 PRS 間で Web ページ収集情報の交換を行うものとする．

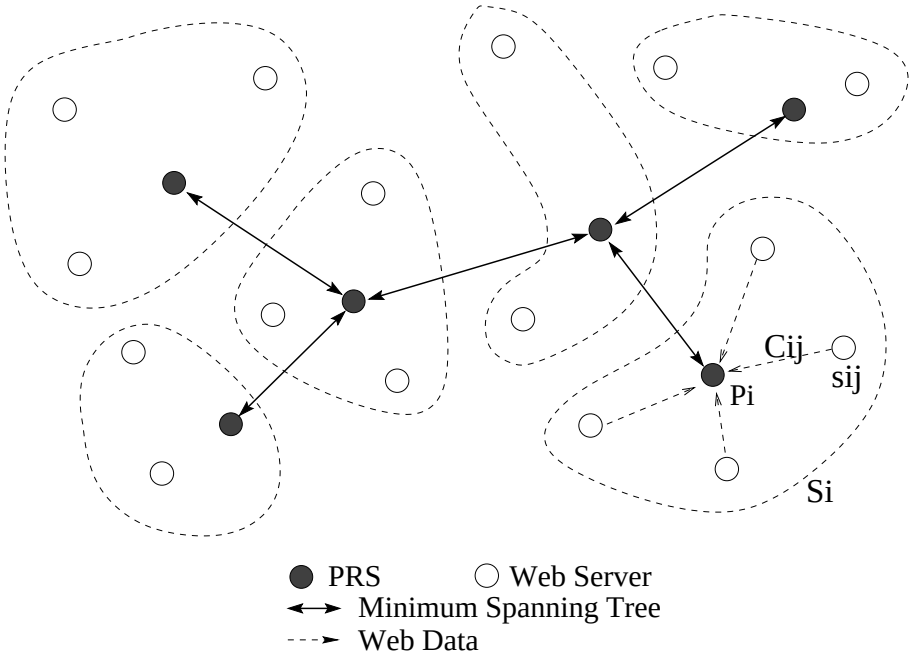


図 6: 最小木に沿った情報交換

分散収集の手順

分散収集の手順を図7に示す．初期状態においては，どのサーバをどのPRSに割り当てればよいのかわからないため，ランダムに割り当てる．その割り当てのもとでWebページを収集し，次節に示すコスト均等化アルゴリズムにより隣接PRS間でコストの均等化を行う．その結果新たに得られた割り当てリストのもとで再びWebページを収集する．以上の一連の動作を繰り返すことにより，ネットワーク状態の変化に対する柔軟なコスト割り当ての実現を試みる．

以下，コスト均等化アルゴリズムについて説明する．

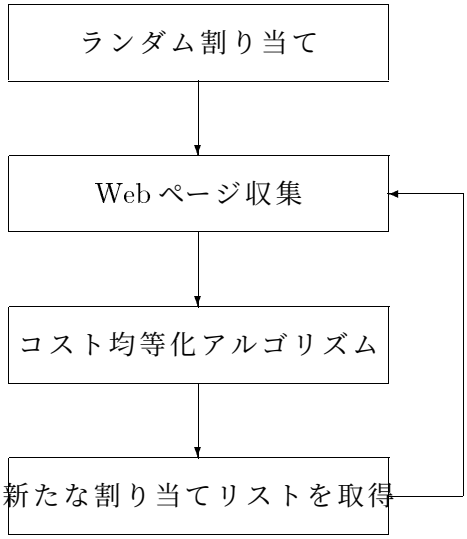


図 7: 分散収集の手順

コスト均等化アルゴリズム

コスト均等化アルゴリズムは以下の手順からなる．

1. 隣接PRSに対し，移動させたいコスト量$Give$を求める．
2. 互いの$Give$の値を比較して，実際に移動させるコスト量$Move$を求める．
3. サーバとPRSとの距離を考慮して，渡すサーバを決定する．
4. $Move$の値にしたがってサーバの受け渡しを実行する．

手順１

PRS AからPRS Bに移動させたいコスト量を$Give(A, B)$と表す．PRS Aに隣接しているPRSが$B_1, B_2, ..., B_n$のn個であるとき，(3)式で計算される．

$$Give(A, B_i) = \frac{Cost(A) + \sum_{j=1}^{n} Cost(B_j)}{n+1} - Cost(B_i) \tag{3}$$

なお，$Give(A, B)$の値が正ならばサーバを渡し，負ならばサーバを受け取ることになる．

手順２

$Move(A,B)$ は，PRS A からPRS B へ実際に移動させるコスト量を表し，互いの $Give$ の値を比較することで以下のように計算される．

- $Give(A,B) > 0, Give(B,A) > 0$，または $Give(A,B) < 0, Give(B,A) < 0$ のとき
  $Move(A,B) = Give(A,B) - Give(B,A)$
- $Give(A,B) > 0, Give(B,A) < 0$ のとき
  $Move(A,B) = \min\{|Give(A,B)|, |Give(B,A)|\}$
- $Give(A,B) < 0, Give(B,A) < 0$ のとき
  $Move(A,B) = -\min\{|Give(A,B)|, |Give(B,A)|\}$

また，$Move(A,B) = -Move(B,A)$ が成り立つ．

手順１と手順２の実行例

図8を用いて，手順1と2を説明する．各々のPRSに付された数字は，そのPRSの持つWWWデータに対するコストを表すものである．

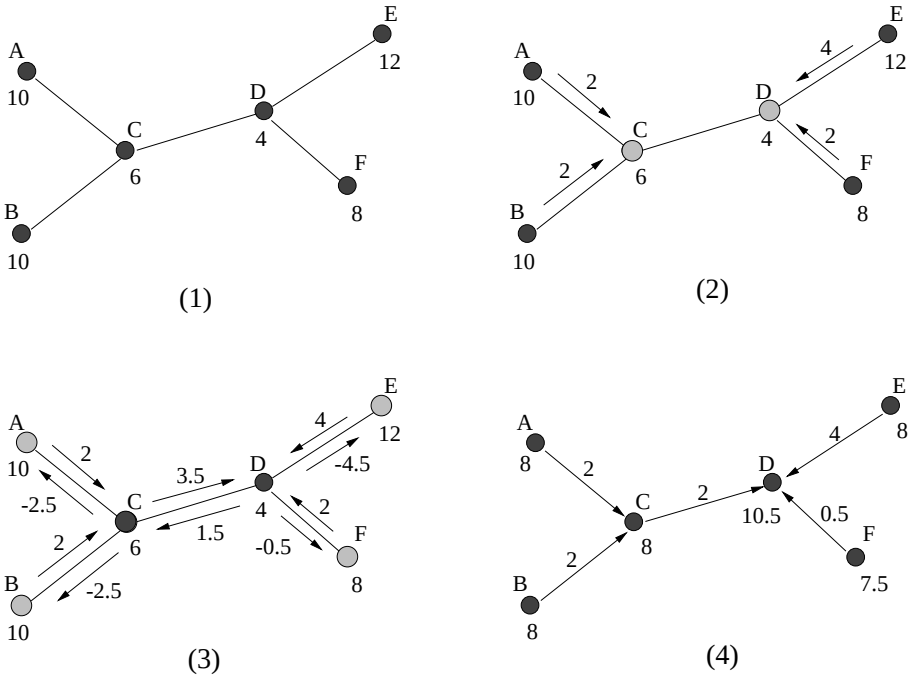


図 8: 均等化の例．

図8（1）のPRS A について考える．A の隣接PRSは C であり，$Cost(A) = 10$, $Cost(C) = 6$ であるから，$Give(A,C)$ は以下のように計算される．

$$Give(A,C) = \frac{10+6}{2} - 6 = 2$$

PRS B，E，F についても同様にして $Give$ の値を求めると図8（2）のようになる．

次にPRS C について考える．隣接PRSは A，B，および D であり，コストはそれぞれ6，10，10，4であるから，

$$Give(C, A) = \frac{6 + 10 + 10 + 4}{4} - 10 = -2.5$$

$$Give(C, B) = Give(C, A)$$

$$Give(C, D) = \frac{6 + 10 + 10 + 4}{4} - 4 = 3.5$$

と求められる．PRS D についても同様にして図8（3）が得られる．

ここで A，C 間を見てみると，$Give(A, C) = 2$，$Give(C, A) = -2.5$ となり

$$Move(A, C) = \min\{|2|, |-2.5|\} = 2$$

$$Move(C, A) = -\min\{|2|, |-2.5|\} = -2$$

となる．

また C，D 間では $Give(C, D) = 3.5$，$Give(D, C) = 1.5$ となっている．したがって

$$Move(C, D) = 3.5 - 1.5 = 2$$

となる．

以上のような操作をすべてのPRSが行うと図8（4）が得られる．PRSに付された数字は均等化後のコスト，枝の重みは $Move$ の値である．ただしここでのコストは移動元のコストを基準にして計算したものである．つまりこの例では A から C へコストが2だけ移動しているが，この2というコストは A にとってのコストであり，実際のコストは新たに収集を行うまで決定されない．

手順3

手順2までは，移動させるコスト量だけに注目していて，どのサーバを受け渡しするかどうかを考慮していない．そこで，手順3ではPRSとサーバとの距離を考慮して，どのサーバを渡すかを決定する．

今，PRS R_i からサーバ s_{ij} までの距離を $d(R_i, s_{ij})$ とする．ネットワーク的な距離には様々な定義方法があるが，本稿では「サーバから1KByteのデータを収集するのに要した時間」で定義する．PRSは収集サーバ s_{ij} のデータ量 w_{ij}（単位はKByte）を得ることも可能である．したがって，$d(R_i, s_{ij})$ は次のようになる．

$$d(R_i, s_{ij}) = \frac{t_{ij}}{w_{ij}} \tag{4}$$

これによりコストは，

$$Cost(R_i) = \sum_{j=1}^{n_i} w_{ij} \cdot d(R_i, s_{ij}) \tag{5}$$

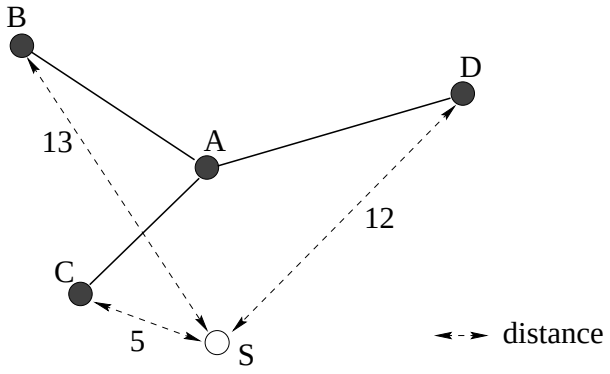


図 9: 隣接 PRS による距離の測定

と表すこともできる.

図 9を考える. PRS A は手順 2 で得られた $Move$ の値にしたがって, 隣接 PRS の B, C, および D のいずれかにサーバ S を渡そうとしている. そこで, 隣接 PRS は S の Web ページの一部を収集し, 式 (4) を用いて S までの距離を算出する. なお本研究ではトップページから 50URL を収集することとした. A は得られた距離が最も短い隣接 PRS に対して S を渡す. この例では C に渡すことになる.

手順 4

手順 2 で得られた $Move$ の値, 及び手順 3 で得られたサーバに基づいて, 実際にサーバの受け渡しの実行を行う.

以上より, 新たな割り当てリストが得られ, その割り当てのもとで再び Web ページの収集を行うことになる.

5.2.2 利用方法

手順 1

まず，Perl が実行できる環境を用意する．以下の説明に登場するディレクトリは全て作業ディレクトリの下にあるものとする．

手順 2

'log/' ディレクトリには各 PRS のログが格納されている．ただし容量が大きいため gzip で圧縮されている．log.pl を実行すると，これらのログを PRS ごとにまとめたファイル（名前は PRS 名と同じ）が作成される．このファイルも'logs/' ディレクトリに置かれる．ファイル形式は「URL ドキュメント量 収集時間」で，URL 順にソートされている．

手順 3

get_distance.pl を実行すると，上で得られたログファイルをもとに PRS から各サーバまでの距離を計算し，'distance/' ディレクトリに PRS の名前でファイルが出力される．ファイル形式は「サーバ名 距離」である．

手順 4

get_document.pl を実行すると，ログファイルをもとに各サーバのドキュメント量が作業ディレクトリの'document.txt' という名前のファイルに出力される．ファイル形式は「サーバ名 ドキュメント量」である．

手順 5

get_cost.pl を実行すると，'distance/' ディレクトリの距離ファイルと document.txt を用いて，PRS ごとに各サーバに対するコストを計算し，'cost/' ディレクトリに PRS の名前でファイルが出力される．ファイル形式は「サーバ名 コスト」である．

手順 6

make_random_list.pl はランダムにサーバを割り当てるために実行するもので，'random/' ディレクトリに PRS の名前で担当サーバリストのファイルが出力される．

手順 7

distribution.pl はコスト均等化アルゴリズムを実行するプログラムで，上で得られた各情報をもとに，コスト均等化を試みる．なお，プログラム上でアルゴリズムの実行回数を任意に設定することが可能である．実行結果は「PRS 名: コスト: 担当サーバ数」の形式で標準出力に出力される．

5.3 PRS 自動バージョンアップ方式

PRS 自動バージョンアッププログラムは，分散型 WWW ロボットのクライアント側，すなわち PRS のバージョンアップを自動化するものである．これによって，ソフトウェアのバージョンアップに伴う作業が大幅に短縮される．

5.3.1 プロトコル仕様

毎日一回[7]，HTTP プロトコルを使い[8]，指定された URL にアクセスし，そこに存在するプログラム (PRS の tar.gz ファイル) が現在ローカル (クライアント) に使っている PRS よりもバージョンが新しい時，自動的に取得し，プログラムを更新する．

本プログラム (`PRS_Update.java`) は，Java(`JDK1.2.1\_03` 以降) で動作する．

1. 設定ファイル 1(update.conf)
   update.conf には以下の内容を記述する．

   prs-update.source PRS のプログラムが格納してある URL を記述

   prs-update.id アクセスのための ID を記述

   prs-update.pass アクセスのためのパスワードを記述

   prs-update.gzip クライアントマシンの gzip のパスを記述

   例

   ```
   prs-update.proxy=proxy.dwr.com:8888
   prs-update.source=http://www.dwr.com/prs/
   prs-update.id=access_id
   prs-update.pass=passwd
   prs-update.gzip=/usr/local/bin/gzip
   ```

2. 起動方法

   ```
   /bin/sh prs-update.sh /data/class/ /data/class/update.conf
   ```

   - prs-update.sh は起動用スクリプトファイル
   - 第 1 引数は JAVA プログラムの CLASSPATH
   - 第 2 引数は 1の設定ファイル

3. 設定ファイル 2(update.date)
   現在クライアントにインストールされている PRS が，最新かどうかをチェックするために，2の第 2 引数で指定されているディレクトリに update.date というファイルを置き，その中で現在使われている

[7] cron により毎日起動する
[8] Proxy 経由も利用可能とする

prs- 西暦 (4 桁)/ 月 (2 桁)/ 日 (2 桁)/ バージョン (1 桁).tar.gz

のファイル名を格納しておく.

例: prs-200001011.tar.gz

4. ログファイル (`prs_update.log`)
   プログラム実行中の一連のログを記録するために, 2の第 2 引数で指定されているディレクトリに`prs_update.log`というファイルを作成し, そこにログを出力する.

5. 起動後の処理

   (a) update.conf から設定内容を読み込む.

   (b) (5a) で, proxy の記述が存在した場合には, proxy の設定をする. proxy サーバーにアクセスするデフォルトの port 番号は 80 である.

   (c) update.date から現在の PRS のバージョンを読み込む. update.date が存在しない場合には, (5d) の動作を行なわずに (5e) で最新のプログラムをダウンロードする.

   (d) prs-update.source に記述されている URL にアクセスし, 最新のバージョンのプログラム名を調べる. これが, (5c) で読み込んだバージョンより新しければ (5e) で, 最新のプログラムをダウンロードする. (5c) で読み込んだバージョンと, 最新のバージョンが一致したときは, (5e) 以下の作業を行なわずに, 処理を終了する.

   (e) prs-update.source に記述されている URL にアクセスし, 最新のバージョンのプログラムをダウンロードする.

   (f) (5e) でダウンロードしたプログラムを gzip で解凍, インストールする. ダウンロードした prs-xxxx.tar.gz ファイルはインストール後, 削除する.

   (g) update.date の内容をダウンロードしたプログラム名に更新する.

   (h) `prs_update.log` にログを書き出す.

   (i) 処理を終了する.

5.3.2 利用方法

以下の説明では,

$WORK 2.(37ページ) の第2引数で指定されているディレクトリ

$START prs-update.sh の置かれているディレクトリ

$CLASSPATH PRS_Update.class の置かれているディレクトリ

とする.

1. 手動で実行する場合
   以下のコマンドを実行する.

   ```
   /bin/sh $START/prs-update.sh $CLASSPATH $WORK/update.conf
   ```

   例

   ```
   $WORK=/data/class/
   $START=/data/
   $CLASSPATH=/data/class/
   ```
   の場合

   ```
   /bin/sh /data/prs-update.sh /data/class/ /data/class/update.conf
   ```

2. cron を使用して自動で実行する場合
   crontab に以下を記述する.
   毎日0時に起動する場合

   ```
   0 0 * * * /bin/sh $START/prs-update.sh $CLASSPATH $WORK/update.conf
   ```

5.4 PRS 収集データ再配布システム

再配布システムはPRS(以下では「WWW データ収集クライアント」と呼ぶ) からデータを一カ所に収集 (このプログラムを「データ収集システム」と呼ぶ) し，第三者に配布する (この配布するためのプログラムを「データ配布システム」，データを受け取るためのクライアントを「第三者クライアント」と呼ぶ).

5.4.1 プロトコル仕様

システム構成図を図 10に示す.

本システムは，WWW データ収集クライアント (PRS) の特定のディレクトリにおかれたファイル (仕様は以下参照) をデータ収集システムに定期的に（1 日 1 回）集め，データ収集システムから第三者クライアントへデータを再配布する.

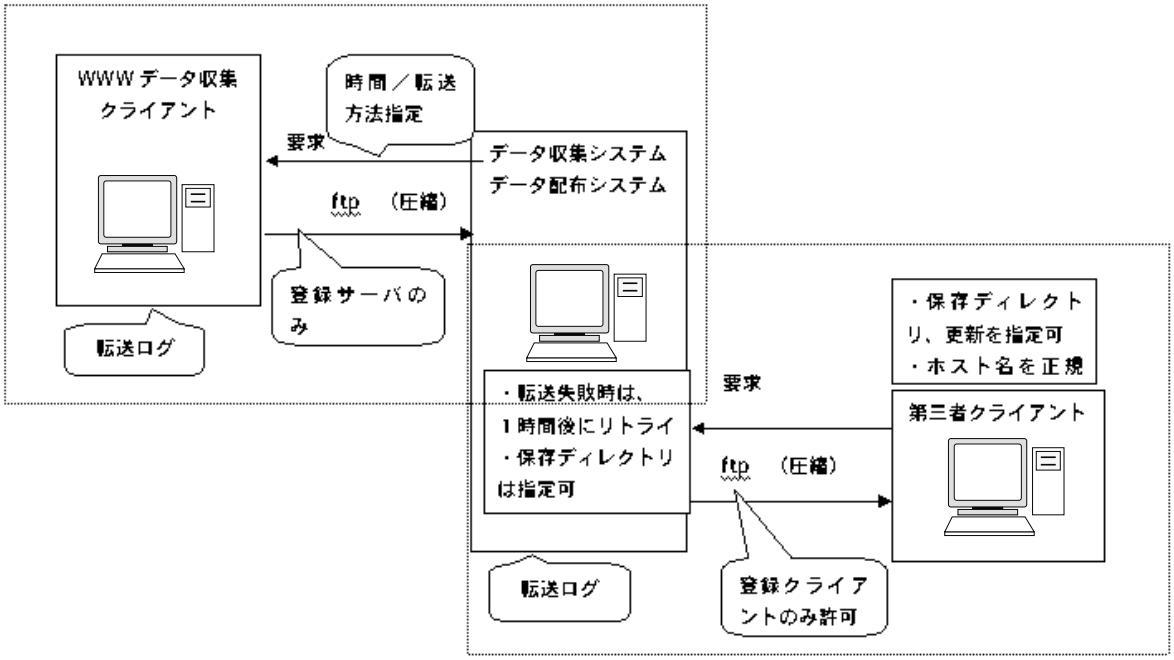


図 10: PRS 収集データ再配布システム

WWW データ収集クライアントは，収集した WWW データ (Web ページ) をデータ収集システムに配送する．WWW データ収集クライアントが Proxy 外 (インターネットに直結) の場合，データ収集システムは WWW データ収集クライアントに直接，接続することができる．データ収集システムからの要求時，WWW データ収集クライアントはデータ収集システムにデータを転送する.

一方，データ収集クライアントが Proxy 内 (ファイアウォール内) の場合，WWW データ収集クライアントはデータ収集システムからスケジュール設定ファイルを取得し，該当のデータをデータ収集システムに転送する.

データ収集システムはスケジュール設定ファイルを読み込んで WWW データ収集クライアントからデータを取得する．スケジュール設定ファイルには，どの WWW

データ収集クライアントに接続するか，収集するデータの条件(日付等)が入る．

データ配布システムは，要求によって，第三者クライアントにデータを配布する．第三者クライアントの環境ファイルに要求する情報が入れる．

第三者クライアントはデータ配布システムから，WWWデータを要求するエンドユーザである．要求はデータ配布システムに送信される．第三者クライアントのデータ管理機能は本システムの対象外である．

「WWWデータ収集クライアント」→「データ収集システム」

1. 「データ収集クライアント」から「データ収集システム」へは，「データ収集システム」からの要求に基づいてファイルを転送する．セキュリティ確保のため，あらかじめ「データ収集クライアント」側に登録してある「データ収集システム」にしかデータを送れない．

2. 転送時には，データをzipにより圧縮して転送する．

3. ログ（転送開始時間，終了時間，転送速度，転送ファイル容量，総数）を「データ収集クライアント」側に残す．ログは，１週間毎別ファイルとして保存する．

「データ収集システム」

1. 「データ収集クライアント」毎に，収集開始の要求を出せる．

2. 上記のスケジューリングをファイル(クライアント名，開始時間，転送ファイル条件)を指定可能．開始時間は，UNIXのcrontabの書式に準拠する．転送ファイル条件には，全転送，差分転送，特定日指定転送（特定日を指定した場合は，その特定日以降のデータのみを転送），指定期間転送（いつからいつまでを指定）をクライアント毎に指定可能．

3. 「データ収集システム」側が持っている転送ログと突き合わせ，差分転送であれば，差分のみの転送要求を「WWWデータ収集クライアント」に出す．この時，クライアント側には，XXXX/XX/XX～YYYY/YY/YYというように更新日の日付を送る．

4. 「WWWデータ収集クライアント」からのデータ転送ログ（クライアント名，転送開始時間，終了時間，転送ファイル容量，総数）を「データ収集システム」側に残す．ログは１週間毎別ファイルとして保存する．

5. 転送にfailした場合は，１時間後にリトライを行い，それでも失敗した場合には，上記でスケジューリングされる日に再トライする．

6. 転送データは，「データ収集システム」側の指定されたディレクトリに保管される．

7. 再配布効率化のために，一日毎の差分ファイルは１週間分保存する．

「データ配布システム」→「第三者クライアント」

1. 第三者クライアントからの要求に基づいてデータを転送する.
2. 転送時には，データを zip で圧縮して転送する.
3. 認証を行い「データ配布システム」側へ登録してある「第三者クライアント」のみにしか転送できない.

「第三者クライアント」→「データ配布システム」

1. 転送要求は，(ホスト名, 更新日, 指定期間) の指定が可能．また，ホスト名は正規表現で記述可能.
2. 「第三者クライアント」側では，取得データを格納するディレクトリを指定できる．指定ディレクトリ以降の構造は，以下のディレクトリ構成と同様であり，hostname:port 番号.zip ファイルで保存.

以下に，WWW データ収集クライアントのデータ格納ディレクトリの仕様を示す.

WWW データ収集クライアントのデータ格納ディレクトリ

`html/[sumh]/[suml]/[servername:port]/`

[sumb] [suml] は，servername:port に対して

```
static String createDataDirectory(String base, String server) {
    int sum = 0;
    for (int i = 0;  i < server.length();  i ++) {
        sum = sum*13+server.charAt(i);
    }
    String sumh = Integer.toString(sum & 0xff00, 16);
    String suml = Integer.toString(sum & 0x00ff, 16);
```

という計算をした結果.

ディレクトリ内のファイル構成

1. **extserver.list**
   このサーバから外のサーバへのリンク情報.
   1 行が「[リンク先 URL] [TAB] [リンク元 URL]」になっている.

2. **transferedurl.list**
   このサーバ内の，転送したURLのリスト．1行に1エントリ．サーバ名は書かない．http://www.foo.jp/ だったら，「/」と書く．

3. **transferedurldate.list**
   transferedurl.list の1エントリが 24 Byte に対応する．

   最初の 8Byte: 転送した日付 (1970年1月1日00:00 GMT からの経過ミリ秒)

   次の 8Byte: データの最終更新日 (情報がなければ0) (1970年1月1日00:00 GMT からの経過ミリ秒)

   次の 8Byte: データサイズ [Byte]

5.4.2 利用方法

インストール

1. **WWW データ収集クライアントのインストール**

   Redistribution.zip をハードデイスクの任意デイレクトリに解凍し,

   ```
   sh makeFileDataCollectionClient.sh
   ```

   を実行する.

2. **データ収集システムとデータ配布システムのインストール**

   Redistribution.zip をハードデイスクの任意デイレクトリに解凍し,

   ```
   sh makeFileaCollectDistributeServer.sh
   ```

   を実行する.

3. **第三者クライアントのインストール**

   Redistribution.zip をハードデイスクの任意デイレクトリに解凍し,

   ```
   sh makeFileThirdParty.sh
   ```

   を実行する.

システムの起動

1. **WWW データ収集クライアントの起動**

   WWW データ収集クライアント (Proxy なし) を起動するためには, 下記のいずれかのコマンドを使用する.

   1. java -DENV_FILE_PATH= 環境ファイルパス名
      Redistribution.prsClient.PRSClient

   例:

   ```
   java -DENV_FILE_PATH=(environment file path name)
           Redistribution.prsClient.PRSClient
   ```

   2. java Redistribution.prsClient.PRSClient

WWW データ収集クライアント (Proxy あり) を起動するためには，下記のいずれかのコマンドを使用する.

1. java -Dhttp.proxyHost = IPAddress -Dhttp.proxyPort=PortNo
   -DENV_FILE_PATH=(environment file path name)
   Redistribution.prsClient.PRSClient

例:

java -Dhttp.proxyHost = 10.7.5.121 -Dhttp.proxyPort=1080
-DENV_FILE_PATH=/japan/datacollection/WWWCollectionClient.rc
Redistribution.prsClient.PRSClient

2. java -Dhttp.proxyHost = IPAddress -Dhttp.proxyPort=PortNo
   Redistribution.prsClient.PRSClient

例:

java -Dhttp.proxyHost = 10.7.5.121 -Dhttp.proxyPort=1080
Redistribution.prsClient.PRSClient

”環境ファイルパス名”は絶対パス，または相対パスである (クラスがインストールされたデイレクトリに相対する). 2. で，環境ファイルパス名が指定されていない場合，ファイル名”WWWGatherClient.rc”(デフォルトの WWW データ収集クライアントの起動用環境ファイル) が必要.

2. データ収集システムの起動

下記のいずれかのコマンドを使用して，WWW データ収集システムを起動する.

1. java -DENV_FILE_PATH= 環境ファイルパス名
   Redistribution.collectDistribute.DataCollectionServer

   例

   java -DENV_FILE_PATH
   JAPAN/myProjects/NTTSoft/CollectionServerFiles/CollServerEnv.list
   Redistribution.collectDistribute.DataCollectionServer

2. java Redistribution.collectDistribute.DataCollectionServer

“環境ファイルパス名“は絶対パス，あるいは相対パスである (クラスがインストールされたデイレクトリに相対する). 2. で環境ファイルパス名が指定さ

れていない場合，ファイル名“WWWGatherServer.rc”のファイルが必要となる．これはデータ収集システム起動用のデフォルト環境ファイルである．

3. **データ配布システムの起動**

下記のいずれかのコマンドでデータ配布システムを起動する．

1. java -DENV_FILE_PATH=environment file path name
   Redistribution. collectDistribute.DataDistributionServer

例

```
java  -DENV_FILe_PATH
      JAPAN/myProjects/NTTSoft/CollectionServerFiles/distributeEnv.list
      Redistribution. collectDistribute.DataDistributionServer
```

2. java Redistribution. collectDistribute.DataDistributionServer

“環境ファイルパス名“は絶対パス，あるいは相対パスである (クラスがインストールされたデイレクトリに相対する)．2. で環境ファイルパス名は指定されていない場合，“WWWDeliveryServer.rc”(デフォルトのデータ配布システム起動用の環境ファイル) が必要．

4. **第三者クライアント**

下記のいずれかのコマンドを使用して，第三者クライアント (Proxy なし) を起動する．

1. java -DENV_FILE_PATH= 環境ファイルパス名
   Redistribution.thirdParty.ThirdPartyClient

例

```
java  -DENV_FILE_PATH
      JAPAN/myProjects/NTTSoft/CollectionServerFiles/distributeEnv.list
      Redistribution.thirdPartyClient ThirdPartyClient
```

2. java Redistribution.thirdParty.ThirdPartyClient

下記のいずれかのコマンドを使用して，第三者クライアント (Proxy で) を起動する．

1. java -Dhttp.proxyHost = IPAddress -Dhttp.proxyPort=PortNo
   -DENV_FILE_PATH=(environment file path name)
   Redistribution.thirdParty.ThirdPartyClient.

例

```
java -Dhttp.proxyHost = 10.7.5.121  -Dhttp.proxyPort=1080
     -DENV_FILE_PATH=/japan/redistribution/WWWDeliveryClient.rc
      Redistribution.thirdParty.ThirdPartyClient.
```

2. java -Dhttp.proxyHost=IPAddress -Dhttp.proxyPort=PortNo
   Redistribution.thirdParty.ThirdPartyClient

例:

```
java -Dhttp.proxyHost = 10.7.5.121  -Dhttp.proxyPort=1080
     Redistribution.thirdParty.ThirdPartyClient.
```

“環境ファイルパス名“は絶対パス，あるいは相対パスである (クラスがインストールされたデイレクトリに相対する). 2. で環境ファイルパス名が指定されていない場合，“WWWDeliveryClient.rc”(第三者クライアント起動用のデフォルトの環境ファイル) が必要.

環境設定ファイルの設定については，プログラムに添付されるマニュアルを参照されたい.

6 性能評価実験

本章では，分散協調サーチロボットの性能評価実験についてまとめる．

6.1 実験参加サイトと対象 WWW サーバ

PRS をインストール済 (2000 年 2 月末時点) のサイトを表 2に示す．性能評価実験は，これらのサイトの内，表 3に示す 17 台の PRS(Public Robot Server) を用いて行った[9]．また，実験対象として選んだ WWW サーバは，表 4に示す日本国内の 6,500 サーバである．これらの WWW サーバはこれまでの実験で把握している 50,000 サーバからランダムに抽出したものである[10]．

表 2: PRS をインストール済のサイト一覧 (順不同)

設置場所
大学 (17)
慶応義塾大学，琉球大学，奈良先端大学院大学，広島大学， 東京工業大学，南山大学，岐阜大学，東京大学，徳島大学， 京都大学，北陸先端科学技術大学院大学，岩手県立大学， 横浜国立大学，早稲田大学，九州大学，筑波大学， 大阪教育大学
ＩＳＰ (4)
Intio プロバイダ, (株) ネットウェーブ四国， トライネットワークインターナショナル（株）， ネスク・インターネット
国立研究所 (3)
農林水産省 農業研究センター，通商産業省 電子技術総合研究所， 学術情報センター
企業 (2)
ＮＴＴ未来ねっと研究所，シャープ株式会社
ネットワーク関連団体 (3)
東北インターネット協議会，北海道地域ネットワーク協議会， 東海地域ハブ研究会

[9] ネットワークやシステムの異常等による動作停止がなく，全ての実験において問題なく動作したサイトのみを抽出した結果 17 台の PRS となった．

[10] 実験時間短縮のため，6,500 サーバのみを用いて実験を実施．

表 3: 評価実験に用いた PRS(順不同)

	設置場所
PRS01	慶応大学
PRS02	Intio プロバイダ
PRS03	農林水産省 農業研究センター
PRS04	琉球大学
PRS05	奈良先端大学院大学
PRS06	ＮＴＴ未来ねっと研究所
PRS07	東北インターネット協議会
PRS08	広島大学
PRS09	東京工業大学
PRS10	通商産業省 電子技術総合研究所
PRS11	南山大学
PRS12	岐阜大学
PRS13	東京大学
PRS14	徳島大学
PRS15	京都大学
PRS16	北海道地域ネットワーク協議会
PRS17	学術情報センター

表 4: 評価実験の対象とした WWW サーバの分類

ドメイン	サーバ数
ac.jp	2,580
co.jp	2,371
or.jp	606
ne.jp	435
go.jp,kek.jp	137
gr.jp	53
ad.jp	27
地域名.jp	291
合計	6,500

6.2 対象 WWW サーバの特徴解析

対象とした 6,500 の WWW サーバが持つ総 URL 数は，4,653,140URL であり，単純平均で 615URL/Server，メディアンは 82URL/Server であった．また，総データ容量は 22GB，単純平均で 3.5MB/Server，メディアンは 339KB/Server であった．

総容量の分布を図 11に示す．図の横軸は 1 サーバ当りのデータ容量 (KB) であり，縦軸が累積 (サーバ数及び容量) である．図から明らかなように，極端にデータ量が多いサーバが存在し，データ量が多い上位 1 % (65 台) の WWW サーバで総データ量の 43 % を占めている．すなわち，WWW サーバのデータを高速に収集するためには，これら大容量のデータを持つ WWW サーバに対して複数の PRS により並列収集することが重要となる．

また，URL 数の分布を図 12に示す．

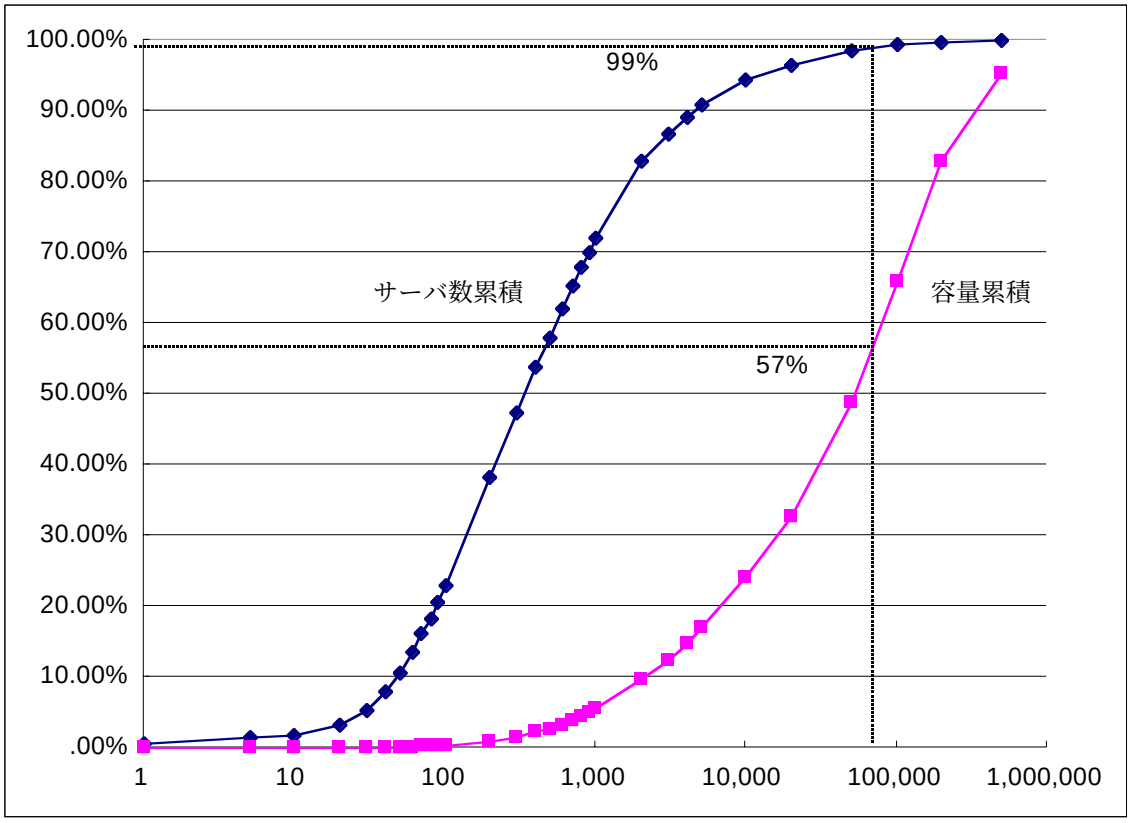


図 11: 各 WWW サーバのデータ量の分布

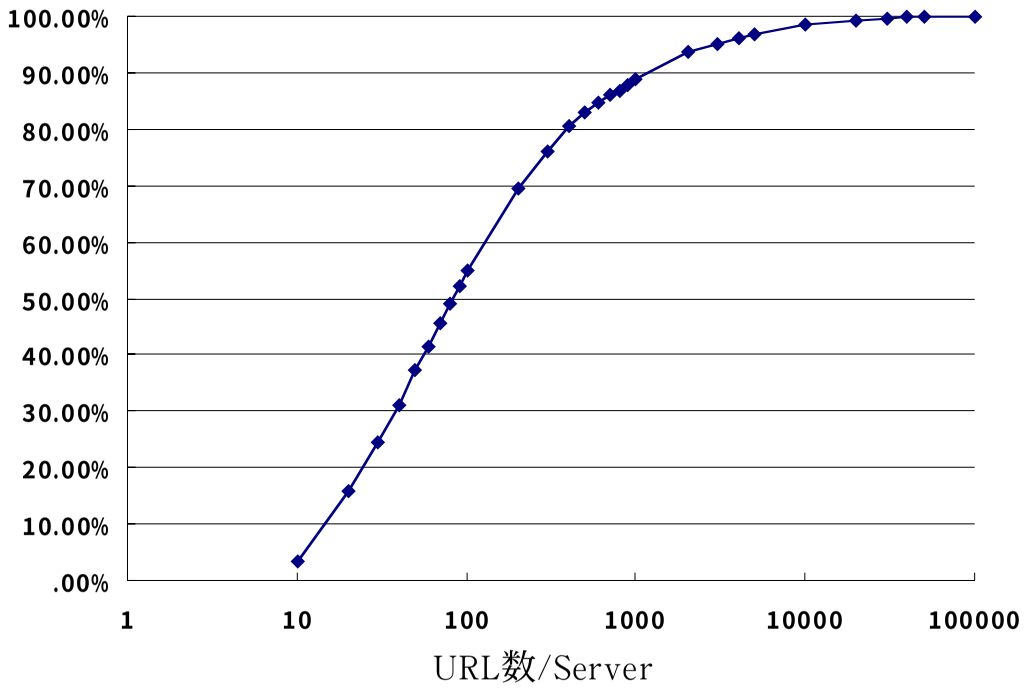


図 12: 各 WWW サーバの URL 数の分布

6.3 負荷均等化評価

負荷均等化評価では，まず，17PRS から 6,500 のサーバに対してトップページ (例:`http://www.etl.go.jp:80/`) から 50URL のみの限定収集を行い，各 PRS と 6,500 サーバ間のデータ転送速度 (byte/s) を求めた.

同一の WWW サーバに対するデータ転送速度は，17 個の PRS で大きな差が認められた．一番速い PRS と一番遅い PRS との差は，2 倍〜710 倍，平均 67.5 倍，メディアン 48.2 倍であった．図 13にデータ転送速度差の分散を示す．上記より，ランダム分散では，大きな負荷の不均衡が発生することが予測され，負荷の均等化が必要になることがわかる.

また，表 5に各 WWW サーバについて 17PRS 内で最高速・最低速のデータ転送速度を持つ WWW サーバ数を各 PRS 毎に集計したサーバ数を示す．例えば，北海道地域ネットワークは，多くの WWW サーバに対して最低速となっているが一方で 15 の WWW サーバについては最高速を示す．15 の WWW サーバをチェックしたところ，北海道地域ネットワーク内の 1 台の WWW サーバ及び IMnet に接続された go.jp ドメインの 14 台の WWW サーバであった.

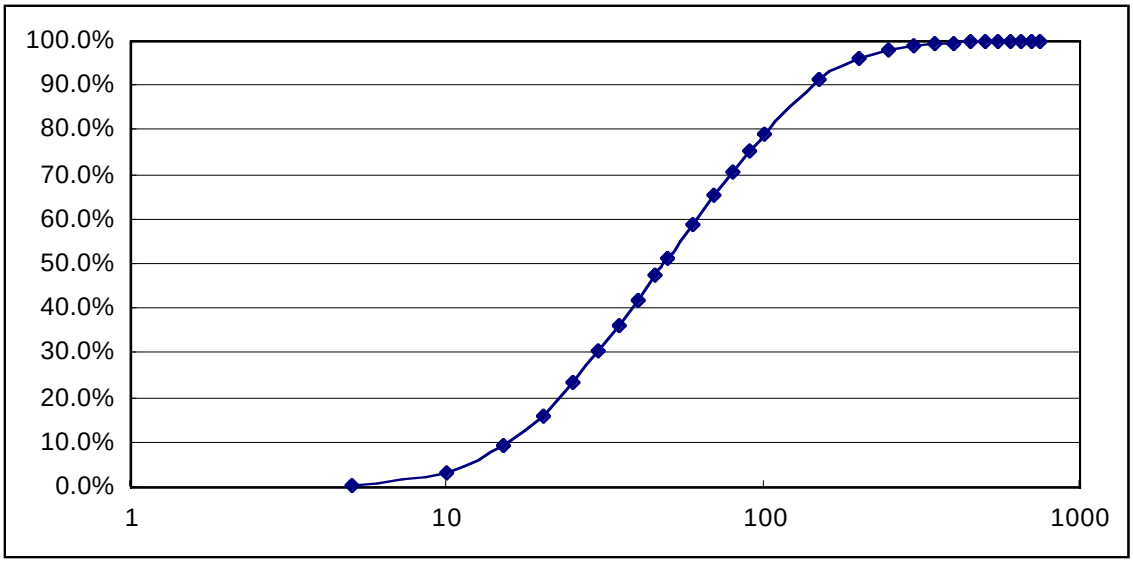


図 13: 17PRS と 6500WWW サーバ間のデータ転送速度差 (倍) の分散

負荷均等化では，5.2で述べたアルゴリズムを用いて各PRSへのスケジューリングを求めた．その結果，表6に示すスケジューリングが得られ，このスケジューリングを用いて評価を行った．

評価では，ランダム分散時に比較して負荷均等化により，どの程度負荷が均等化されるかを調べた．表7に，ランダム分散（予測値）及び均等化分散 (予測値及び実測値) を示す．

表7より，ランダム分散に比較して負荷均等化を行うことにより，実測で 11.1 倍の速度向上が得られることがわかる．予測値と実測値の間の誤差は平均 25 %であり，ほぼ予測通りに負荷均等化の効果が得られた．

表 5: 各 WWW サーバについて 17PRS 内で最高速・最低速のデータ転送速度を持つ WWW サーバ数

PRS 名	最低速	最高速
PRS01	3	94
PRS02	3	14
PRS03	7	191
PRS04	19	314
PRS05	1	225
PRS06	6	3,040
PRS07	0	152
PRS08	2	290
PRS09	4	200
PRS10	425	0
PRS11	3	76
PRS12	4	229
PRS13	5	336
PRS14	4	114
PRS15	2	1,101
PRS16	6,004	15
PRS17	8	40
合計	6,500	6,501(*)

(*) 同一最高転送速度を 2PRS が持つ.

表 6: 負荷均等化分散時の各PRSの割当WWWサーバ数

	設置場所	割り当てられたWWWサーバ数
PRS01	慶応大学	397
PRS02	Intioプロバイダ	739
PRS03	農林水産省 農業研究センター	417
PRS04	琉球大学	388
PRS05	奈良先端大学院大学	390
PRS06	ＮＴＴ未来ねっと研究所	52
PRS07	東北インターネット協議会	392
PRS08	広島大学	479
PRS09	東京工業大学	433
PRS10	通商産業省 電子技術総合研究所	35
PRS11	南山大学	375
PRS12	岐阜大学	471
PRS13	東京大学	466
PRS14	徳島大学	376
PRS15	京都大学	393
PRS16	北海道地域ネットワーク協議会	318
PRS17	学術情報センター	379

表 7: 均等化分散の効果 (単位:hour)

	ランダム分散 (予測)	均等化分散 (予測)	均等化分散 (実測)
PRS01	68.8	41.7	28.2
PRS02	71.1	107.9	104.5
PRS03	88.3	35.0	43.7
PRS04	61.5	55.1	29.6
PRS05	51.7	63.7	53.7
PRS06	33.8	146.2	96.7
PRS07	57.6	55.2	44.7
PRS08	50.8	80.7	71.1
PRS09	43.1	39.0	38.7
PRS10	335.5	32.8	11.0
PRS11	51.8	38.3	28.8
PRS12	63.9	39.3	46.2
PRS13	70.6	56.1	13.7
PRS14	66.4	61.7	40.8
PRS15	37.1	51.4	52.4
PRS16	1216.1	107.9	109.5
PRS17	61.0	61.0	52.1
最大値	1216.1	146.2	109.5
ランダム分散を基準とした時の速度向上	-	8.3	11.1

6.4 分散協調サーチロボット総合評価

単一のPRSを用いて6,500台のWWWサーバの全データを収集する時間は，各PRSによって異なり，各PRSが1スレッドを用い逐次に収集した場合，最小28日(京都大学)～最大712日(北海道地域ネットワーク協議会)と推定される[11].

一方，各PRSが200スレッドで並列に収集した場合でも最小23時間(NTT未来ねっと研究所)～最大23日(北海道地域ネットワーク協議会)が必要となる[12].

表8: 6,500のWWWサーバを収集した時の収集時間(hour)

	1スレッド時	200スレッド時
PRS01	1,008	32
PRS02	1,518	55
PRS03	1,201	55
PRS04	975	32
PRS05	987	24
PRS06	793	23
PRS07	1,066	59
PRS08	953	27
PRS09	1,033	34
PRS10	6,121	244
PRS11	1,158	42
PRS12	904	26
PRS13	1,025	31
PRS14	904	26
PRS15	691	26
PRS16	17,085	544
PRS17	1,663	53
均等化分散	109.5	1.9

表8 [13]及び図14に1スレッド及び200スレッド時の収集時間を示す．これを負荷均等化分散によって，高速化することにより，図15に示すように，17台のPRSで6.3倍(NTT未来ねっと研究所を基準)～286倍(北海道地域ネットワークを基準)の高速化を達成することができた.

本結果から，もともと高速に収集可能な計算機を基準とした場合には，17台に分散しても，必ずしも17倍の高速化が達成できないことがわかる．これは，例えば，処理能力100の計算機と処理能力50の計算機を使って並列計算した時に最大でも

[11] 17PRSから6500のWWWサーバに対する転送速度を50URLのみのページを収集することによって求め，Σ {(WWWサーバが持つデータ容量)÷(転送速度)} により逐次で収集した場合の実行時間を算出.

[12] MAX {((1スレッドでの収集時間)÷200),MAX(PRSが担当する各WWWサーバの収集時間)} により算出した．各PRSで200スレッドに最適な分散ができたと仮定.

[13] 掲載の数値はデータ転送に必要な時間であり，PRSの仕様である「接続切れ時の待ち時間(20秒)」は含まれない.

1.5 倍にしかならないことと同等である[14].

以上から，分散型WWW ロボットを使って高速な収集を実現するためには，ネットワーク的に高速収集が可能な地点にPRS を配置することが効率化の点で重要であることが数値的に証明された.

なお，現在各PRS では200 のサーバに対して同時にアクセスできる仕組みを持っており，この200 スレッドを効率よく利用する均等化アルゴリズムについても，今後さらなる検討が必要である．すなわち，200 スレッドを並列に動作させる場合には，収集に時間のかかるWWW サーバの収集を先行させて開始させ，負荷均等化を図る必要がある.

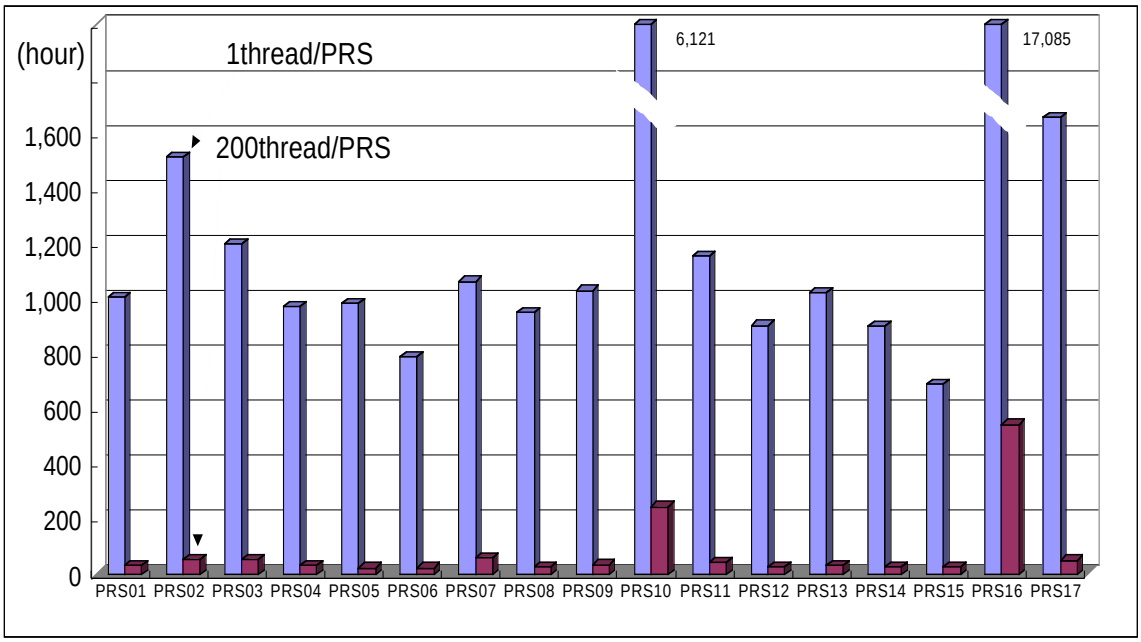


図 14: 1PRS で6500WWW サーバの全データを収集した場合の時間

[14] ここでの処理能力がPRS のデータ転送速度に対応する.

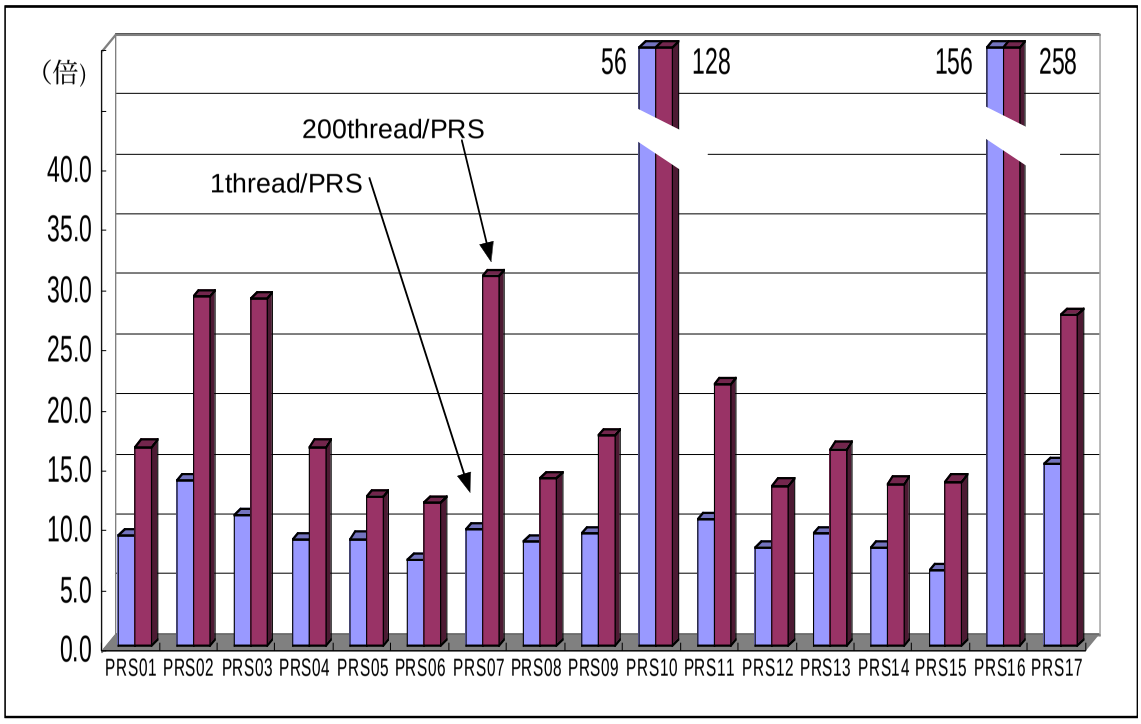


図 15: 分散収集による速度向上率 (倍)(各 PRS で全データを収集した場合を基準)

7 外部発表

7.1 インターネットテクノロジー・ワークショップ'99

1999年8月25日～27日に開催された，日本ソフトウェア科学会主催の「The Second Workshop on Internet Technology」において，以下の研究発表 [14] [8] を行った.

黒田洋介，村岡洋一: WWW上のテキストデータの収集を支援する大容量公開キャッシュサーバの設計と実装, The Second Workshop on Internet Technology (1999.8.25-27)

森 英雄，河野 浩之: 実測データに基づく分散協調型WWWデータ収集アルゴリズムの性能評価, The Second Workshop on Internet Technology (1999.8.25-27)

7.2 第18回IPA技術発表会

1999年10月13日～14日に開催された，IPA主催の「第18回IPA技術発表会」において，以下の研究発表 [11] を行った.

村岡, 田村, 山名, 河野, 森, 浅井, 西村, 楠本, 篠田：Internet広域分散協調サーチロボットの研究開発, IPA第18回技術発表会論文集, pp.71-78 (1999.10.13-14)

7.3 インターネットコンファレンス'99

1999年12月15日～16日に開催された，日本インターネット協会，日本ソフトウェア科学会，日本UNIXユーザ会， WIDEプロジェクト共同主催の「インターネットコンファレンス'99」において，以下の研究発表 [18] を行った.

山名：分散型WWWロボットの実験状況と今後の課題, インターネットコンファレンス'99論文集, p.141 (1999.12.15-16)

7.4 講習会：情報の高度利用技術

2000年1月26日～27日に開催された，システム制御情報学会主催の「講習会：情報の高度利用技術～21世紀の情報社会を生きる知恵～」において，以下の紹介を行った.

河野: Web検索技術, システム制御情報学会講習会資料「情報の高度利用技術」(2000.1.26-27)

7.5 北海道地域ネットワーク協議会シンポジウム2000

2000年3月6日～7日に開催された北海道地域ネットワーク協議会主催の北海道地域ネットワーク協議会シンポジウム'2000において，以下の招待講演 (論文) [19] を行った.

山名: 広域分散コンピューティングの現状と課題－分散型WWWロボットを例にとって－,Northシンポジウム'99論文集 (1999.3.6-7)

7.6 岩波講座 マルチメディア情報学

以下の本 [6] で本実験の紹介を行った.

有木, 上原, 田中, 加藤, 河野, 西尾: 情報の構造化と検索, 岩波講座 マルチメディア情報学,ISBN4-00-010968-5 C3355 (2000.3.27)

8 おわりに

8.1 まとめ

本年度の研究開発では，(1) 分散協調サーチロボットの開発，(2) 分散協調サーチロボット環境の実装，(3) 分散協調サーチロボットの評価，の 3 点を行った．

17 分散された分散型 WWW ロボットを用い，日本国内の 6,500 の WWW サーバを対象とした収集実験を行い，実際の分散協調サーチロボットの有効性を確認した．この結果，一カ所で集中して収集する場合に比較し，負荷均一化を行った場合，6.3 ～ 286 倍の高速化が可能であることがわかった．

さらに，PRS プログラムの自動更新システム，及び，収集された WWW データの自動再配布システムの構築を行い，平成 12 年 3 月末からの実運用のための環境を整えることができた．

当初予定していた研究開発のほぼ全てを平成 10 年度～平成 11 年度の 2 年間で達成することができた．しかし，今後の課題で述べるように，インターネット自体が不安定なものであるのに加え，インターネット上に分散された複数のコンピュータの稼働率を 100 %に保つことは現実的に不可能であるため，速度向上率等の厳密な評価において課題を残した．

8.2 今後の課題

8.2.1 今年度予定研究開発項目の一部未達成部分

研究開発項目「分散協調サーチロボットの評価」において，日本国内に配置された 30 以上の分散協調サーチロボットを用いて評価を行うことを当初の予定としていたが，3.3 に示した理由により，評価は 17 個所のみでの評価に留まった．

8.2.2 今後の研究開発に委ねられた課題

インターネットに広域に分散したシステムは，インターネットの接続の不確実性，及び，複数システムの安定動作の確保の問題から，常に全てのシステムを動作させることは極めて困難となる．このため，(1) どのようにして広域に分散したシステムのメンテナンスを行っていくか，また，(2) そのような状況でどのような評価をすべきか，さらに，(3) 負荷の均等化をさらに進めるための研究開発，の 3 点が本研究の実運用を開始するにあたって残された課題である．

8.2.3 研究開発成果の普及方策

本研究開発の成果により，WWW サーバ上のデータを高速に収集することができるようになると共に，インターネット及び WWW サーバに与える負荷を大きく軽減することが可能となる．

このため，本研究開発成果の普及にあたっては，多くの検索サービス提供者との連携が不可欠となる．国内で検索サービスを提供している大学（早稲田大学，京都大学，徳島大学），及び，企業（ＮＴＴ）には既に本実験に参加してもらっており，

今後は一般の商用の検索サービス提供者との連携を積極的に展開していくことを予定している．さらに，システムの運営のためには，ネットワークプロバイダ（既に４社が実験に参加）との連携も強化し，ビジネスモデルを構築していかなければならない．

このような過程を経て，国内での実運用サービスを平成12年度から開始していく予定である．

参考文献

1) I.Foster, C.Kesselman: *The Globus Project: A Status Report*, Proc. of IPPS/SPDP'98 Heterogeneous Computing Workshop, pp.4-18 (1998)

2) S.Kamei,H.Kawano and T.Hasegawa: *Effectiveness of Cooperative Resource Collecting Robots for Web Search Engines*, Proc. of Pacrim'97, Vol.1, pp.410–413 (1997)

3) Steve Lawrence, C. Lee Gilesd: *Searching the World Wide Web*,Vol,280, No.5360, Issue 3, pp. 98 - 100 (1998.4)

4) S.Lawrence and C.L.Giles: *Accessibility of Information on the Web*, Nature, Vol.400, pp.107-109 (1999.7.8)

5) Hayato YAMANA, Kent TAMURA, Hiroyuki KAWANO, Satoshi KAMEI, Masanori HARADA, Hideki NISHIMURA, Isao ASAI, Hiroyuki KUSUMOTO, Yoichi SHINODA and Yoichi MURAOKA: *Experiments of Collecting WWW Information using Distributed WWW Robots*, Proc. of SIGIR'98, Melbourne, Australia, pp.379-380, (1998.8.24-28)

6) 有木, 上原, 田中, 加藤, 河野, 西尾: 情報の構造化と検索, 岩波講座 マルチメディア情報学,ISBN4-00-010968-5 C3355 (2000.3.27)

7) 茨木 俊秀: "アルゴリズムとデータ構造", 昭晃堂 (1989)

8) 黒田, 村岡: *WWW*上のテキストデータの収集を支援する大容量公開キャッシュサーバの 設計と実装, 日本ソフトウェア科学会 The Second Workshop on Internet Technology (1999.8.25-27)

9) 田村健人: 分散 *WWW*ロボットに関する研究, 早稲田大学修士論文, 早稲田大学 (1997.3)

10) 村岡, 河野, 田村, 山名: *Internet*広域分散協調サーチロボットの研究開発, IPA 第17回技術発表会論文集, pp.222-224 (1998.10.21-22)

11) 村岡, 田村, 山名, 河野, 森, 浅井, 西村, 楠本, 篠田： *Internet*広域分散協調サーチロボットの研究開発, IPA 第18回技術発表会論文集, pp.71-78 (1999.10.13-14)

12) 村岡: 「*Internet*広域分散協調サーチロボットの研究開発」研究成果報告書, 情報処理振興事業協会 (IPA) (1999.2)

13) 森 英雄: 分散協調型 *Web*データ収集アルゴリズムの性能評価, 京都大学 情報学科 特別研究報告書 (1999.2)

14) 森 英雄, 河野 浩之: 実測データに基づく分散協調型 *WWW*データ収集アルゴリズムの性能評価, 日本ソフトウェア科学会 The Second Workshop on Internet Technology (1999.8.25-27)

15) 山名早人: *WWW*情報検索の現状, コンピュータソフトウェア, Vol.14, No.5, pp.67–74 (1997)

16) 山名早人, 田村健人, 河野浩之, 亀井聡, 原田昌紀, 西村英樹, 浅井勇夫, 楠本博之, 篠田陽一, 村岡洋一 : 分散型 *WWW*ロボットによる *WWW*情報収集, データ工学ワークショップDEWS98, No.24 (1998.3.5-7)

17) 山名, 田村, 森, 河野, 黒田, 西村, 浅井, 楠本, 篠田, 村岡: 国内の全 *WWW*データを *24*時間で収集する分散型 *WWW*ロボットの試み ,North シンポジウム'99 論文集 (1999.3.6-7)

18) 山名：分散型 *WWW*ロボットの実験状況と今後の課題, インターネットコンファレンス'99 論文集, p.141 (1999.15-16)

19) 山名: 広域分散コンピューティングの現状と課題－分散型 *WWW*ロボットを例にとって－,North シンポジウム 2000 論文集 (2000.3.6-7)