**研究課題** グリッドアプリケーションの研究と開発
**種別** 指定研究
**代表者** 武田俊之(情報メディア教育センター)
**研究員** 地道正行(商学部)、岡田孝(理工学部)

## 1. 概要

プロジェクト２年目の今年度は昨年の成果の上の継続と、さらに新しいソフトウェアの導入とその上での開発をおこなった。

1. 2.　実行可能型一般化リッジ回帰推定量の正確なモーメントの数値評価(地道正行)
2. 分散共有メモリ（タプルスペース）を利用した簡易分散プログラミング環境 (NetWorkSpaces) の導入とテスト
3. NetWorkSpaces 上でのアプリケーションソフトウェアの構築

以下に 1-3 について報告する。2、3 は主に技術の解説である。4 については本誌掲載の「分散共有メモリを利用した Web マイニングベースの構築」に詳細を記述している。

今年度はプロジェクト２年目で最終年度であり、また教育研究システムでは「研究支援」のシステムを導入しないことが決定されていたために、若干今後の関西学院大学の教育研究環境に資するという意味でのモチベーションに欠ける面があった面は否めない。

## 2. 分散プログラミング環境 (NetWorkSpaces) の導入とテスト

現在稼働中のグリッドシステムは、複数のパラメータセットを利用した統計処理プログラムを並行して実行するために、MPI (Message Passing Interface)を用いた単純なものである。しかし、研究上の（また、ビジネス上の）ニーズとしては、処理結果によって新しいパラメータの設定、使用する分析手法の変更などを可能にすることも求められている。

### 2.1. 分散共有メモリとタプルスペースモデル (協調言語Linda)

私たちが利用しているMPI は並列プログラミングにおける通信ライブラリとしてデファクト・スタンダードの地位を占めている。実装も、MPICH とLAMがあり、サポートする言語もFORTRAN、Cをはじめとして、PythonやPerlなどスクリプト言語など幅広い。しかし、Michael Feldmanが Programming Clusters Just Got Easier “ (http://www.hpcwire.com/hpc/663546.html) において、"It's hard to find a real fan of MPI today. Most people either tolerate it or hate it" と指摘しているように実用的ではあるが、"hard to learn, difficult to program, no allowance for incremental parallelization, doesn't scale easily"などの理由から並列プログラミング普及の敷居を高くしている側面もある。

本プロジェクトの目的の一つは人文・社会科学系の研究者をターゲットにしたグリッドコンピューティング、並列計算のフレームワーク、プログラミング環境を構築し、専門的な知識なしに大規模な計算資源を利用することを可能にすることであり、並列システム理解およびプログラミングの容易性の観点からは MPI を利用することは研究者自らがプログラミングをするには難があると思われた。

そこでより理解がしやすいモデルとプログラミング環境を持つものとして、タプルスペースモデルとその協調言語である Linda について導入を検討した。

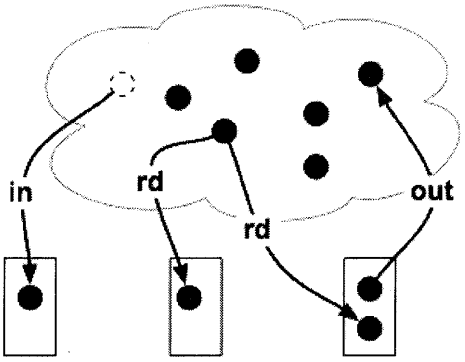
タプルスペースは仮想共有メモリシステムであり、効率的でありながらプログラミングや利用が非常に容易である特徴を持つ。タプルスペースモデルは概念的に理解しやすく、デッドロック、オブジェクトの同期問題、複雑な API といった並列プログラミングの困難さが解消すると考えられる。基本的なプログラミング知識があればグリッドを利用した並列プログラミングを作成することが可能になるであろう。並列性はどこにでもあるものであり、容易にインプリメントすることができる。

タプルスペースはモデルであって、言語やツール、環境そのものではない。タプルスペースの実装と

して、TCP Linda, Paradise, Rinda などが存在する。そのなかでも最も新しい実装が NetWorkSpaces であり、このシステムを試用した。次節で NetWorkSpaces について述べる。

以下にタプルスペースの概略を記す。

タプルスペースモデルはメモリモデルであり、ネットワーク上に存在する仮想の共有メモリ（タプルスペースを介して、ネットワーク上に分散したプロセスがタプル（データ）を読み書きすることによって協調しながらタスクを実行することによって並列処理を遂行する。




左図はタプルスペースモデルを表した概念図である。雲形がタプルスペース、下の矩形がプロセス、黒丸がタプルである。

各プロセスはタプルスペースのデータを読み書きするために、3 つのプリミティブを用いる。

”in”はタプルスペースからデータを取り込む。プリミティブ実行後、タプルスペースにはそのタプルは存在しない。

“rd”はタプルスペースのデータを読む。プリミティブ実行後もタプルスペースにそのタプルは存在する。

“out”はタプルスペースにデータを書き込む。

この 3 つのプリミティブだけでプロセスの強調的なふるまいを記述することができる。また、タプルを”in”した後に”out”することによって、アトミックな更新処理を自然に記述することが可能である。他の並列プログラミングモデルではアトミックな更新処理は非常に困難な部分である。

### 2.2. NetWorkSpaces

NetWorkSpaces (NWS) はスクリプト言語で記述されたプログラムを協調させるためのフレームワークで、タプルスペースモデルに基づいている。NWS は現在 Python、Matlab、R をサポートしている。

NWS はタプルスペースモデルをベースとしたネットワーク上のワークスペース (NetWorkSpace) を介して複数のプロセスが変数—値ペアからなるタプルを読み書きすることによって協調動作する。

NWS ではワークスペース上に存在しない変数を読み出そうとしたとき、その変数がない場合にはエラーを発生させるのではなく、変数に値が割り当てられるまでブロックする。また、ある変数に複数回値が割り当てられたとき、その変数は FIFO (First In First Out) のように動作する。(Last In First Out や変数として動作させることも可能である。たとえば、ワークスペースが ws であるとして、ws.store(a, 1); ws.store(a, 2)を実行した後に、a を読み出すと、1、2 の順で値が返る。

### 2.3. NetWorkSpacesのプログラム例

R から NetWorkSpaces を利用する例を示す。タプルスペースの実体である NetWorkSpaces サーバーはすでに起動済みであるものとする。NetWorkSpaces サーバーの起動法についてはマニュアルを記述されている。

NetWorkSpaces ではオブジェクトはカプセル化されてタプルスペースに書き込まれる。読み込みの場合はその逆である。

また、現在、Web クローリングで取得したデータを形態素解析にかけてデータベース化するプログラムの開発とチューニングしながら、NetWorkSpaces の性能評価と有効性を評価中である。

```
twistd -nyc /usr/local2/etc/nws.tac
```

```
> ws = new("netWorkSpace", "r place")

> cat("connected, listing contents of netWorkSpace (should be nothing there).¥n",
      nwsListVars(ws), "¥n")
connected, listing contents of netWorkSpace (should be nothing there).


> nwsStore(ws, "x", 1)

> cat("should now see x.¥n", nwsListVars(ws), "¥n")
should now see x.
 x        1        0        0        fifo

> cat("nwsFind (but don't consume) x.¥n", nwsFind(ws,
      "x"), "¥n")
nwsFind (but don't consume) x.
 1

> cat("check that it is still there.¥n", nwsListVars(ws),
      "¥n")
check that it is still there.
 x        1        0        0        fifo

> cat("associate another value with x.¥n")
associate another value with x.

> nwsStore(ws, "x", 2)

> cat(nwsListVars(ws), "¥n")
x        2        0        0        fifo

> cat("consume values for x, should see them in order saved.¥n",
      nwsFetch(ws, "x"), "¥n", nwsFetch(ws, "x"), "¥n")
consume values for x, should see them in order saved.
 1
 2

> cat("no more values for x... .¥n", nwsListVars(ws),
      "¥n")
no more values for x... .
 x        0        0        0        fifo

> cat("so try to nwsFetch and see what happens... .¥n",
      nwsFetchTry(ws, "x", "no go"), "¥n")
so try to nwsFetch and see what happens... .
 no go

> cat("create a single-value variable.¥n")
create a single-value variable.


> cat("get rid of x.¥n")
get rid of x.

> nwsDeleteVar(ws, "x")
```

```
> nwsDeclare(ws, "pi", "single")

> cat(nwsListVars(ws), "¥n")
pi          0          0          0          single
x           0          0          0          fifo

> cat("get rid of x.¥n")
get rid of x.

> nwsDeleteVar(ws, "x")

> cat(nwsListVars(ws), "¥n")
pi          0          0          0          single

> cat("try to nwsStore two values to pi.¥n")
try to nwsStore two values to pi.

> nwsStore(ws, "pi", 2.171828182)

> nwsStore(ws, "pi", 3.141592654)

> cat(nwsListVars(ws), "¥n")
pi          1          0          0          single

> cat("check that the right one was kept.¥n", nwsFind(ws,
      "pi"), "¥n")
check that the right one was kept.
 3.141593

> cat("what about the rest of the world?", nwsListWss(ws@server),
      "¥n")
what about the rest of the world?  __default      [system]            False    0
 outer space      IPv4Address(TCP, '127.0.0.1', 59259) (5491)      False    3          input,output,param
>r place          IPv4Address(TCP, '127.0.0.1', 59268) (5491)      False    1          pi
```

**参考文献**

[1] David Gelernter, Generative Communication in Linda, ACM Transactions on Programming Languages and Systems, pp. 80--112, January 1985

[2] Nicholas Carriero and David Gelernter How to Write Parallel Programs - A First Course, MIT Press, 1990

## 3. 実行可能型一般化リッジ回帰推定量の正確なモーメントの数値評価（地道正行）

本研究では, Hoerl & Kennard (1970) によって提案された一般化リッジ回帰(GRR; GRR) 推定量の実行可能型のものである実行可能型一般化リッジ回帰(Feasible GRR; FGRR) 推定量の正確なモーメントを数値的に求めることを目的とした. とくにクロスモーメントの値を具体的に求めることは単一の計算機環境では困難であったため, 収束性などをグリッドコンピューティング環境(Globus Toolkit 3.2, LAM/MPI, R, Rmpi) を用いることによって数値的に評価した. この結果は,Jimichi (2005) にまとめられ, 級数の収束性などのさらに詳しい考察が地道(2007-a) で与えられた. なお, 地道(2007-b) においてこれ

らの研究を通して得られた結果が報告された. 末筆ながら,今回の研究において情報メディア教育センターの武田俊之氏よりグリッドシステムやプログラミングに関するご指導いただいたことに感謝の意を表したい.

## 参考文献

[1] Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: biased estimation for nonorthogonal problems. Technometrics, Vol. 12, pp 55–67.

[2] Jimichi, M. (2005). Improvement of regression estimators by shrinkage under multicollinearity and its feasibility, Osaka University, Ph.D. dissertation.

[3] 地道正行(2007-a). 『実行可能型一般化リッジ回帰推定量の正確なモーメントの数値評価』,情報科学研究, 関西学院大学情報メディア教育センター, Vol. 21, pp 3–22.

[4] 地道正行(2007-b). 『実行可能型一般化リッジ回帰推定量の正確なモーメントについて』, 日本経営数学会, 第 29 回(通算 49 回) 全国研究大会, 報告要旨集, pp 21–26.