米国におけるビッグデータの解析事例

松本　伸哉

米国において，ビッグデータ解析の事例を紹介する．本事例は，オンラインゲームの会社において，ユーザーとの通信をすべて蓄積し，解析を実施した事例である．オンラインゲームにおいては，初心者にとって入り込みやすいゲームであるとともに，熟練したプレイヤーにとっても飽きさせない要素が必要である．また，プレイヤー間でのフェアな競争が必要であり，アンフェアな行為が可能でないかの確認ために，解析を実施した．

キーワード：ビッグデータ，イベントデータ，クリックストリーム，パス解析，Teradata Aster

1. はじめに

今日，コンピュータ関連の技術の発達は目覚ましく，特にデータ蓄積装置であるハードディスクの容量は増加し続けている．安価となったデータ蓄積装置に大量のデータを蓄積し，分析を行うことで価値を生み出すことがわかってきた．1990 年代ごろから，データウェアハウス（データ倉庫）という，時系列に整理された大量の統合業務データ，およびその管理システムが発達してきた．最近は，これまでのデータウェアハウス用の管理システムでは取り扱えない種類のデータが増えてきている．これらを含めたデータをビッグデータと呼んでいる．

ビッグデータは，三つの項目を用いて特徴づけることが多い．一つ目は，「量（Volume）」で，取り扱うデータの総量が，数十テラバイトから数十ペタバイトまでのデータの範囲を対象とすること．二つ目は，「速度（Velocity）」で，1 時間あたり数十ギガバイトから，数十テラバイトまでの範囲を対象とすること．三つ目は，「多様性（Variety）」で，Relational Database Management System (RDBMS) のモデルから，多構造形式のモデルを取り扱うことである．

2. イベント形式時系列データ

時系列データの分析といえば，以前は，月単位の売上げ，気温データなど，等間隔に記録されたデータを用いた分析を指すことが多かった．これは，分析することを目的に収集・蓄積されたデータである．等間隔でデータが記録されているため，微分方程式（差分方程式）によるモデル化を行い，分析が可能である．しかしこれは，分析対象・分析方法を決定した後で，分析のためのデータ収集を行う必要があった．赤池らは，セメント生成用の炉の温度データから，生成の効率・品質を向上させるために赤池情報量基準 (AIC) を開発した [1]．北川らは，経済の時系列分析などに AIC を応用した [2]．

これに対して，何らかのイベントが発生した際に，そのイベント情報が記録される形式のデータがある．このようなデータは，データの収集のためにコストをかけるのではなく，業務で発生するイベント情報を蓄積して分析に使用する．蓄積項目を先に決定しておくと，当初に想定していた分析の枠を超えることは難しい．発生するデータを網羅的に蓄積することにより，当初に想定していた分析の枠を超えて，原因究明のための詳細なドリルダウンなどが実施可能になる．この反面，等間隔データではないため，微分方程式などによるモデル化は困難である．さらにビッグデータ特有の問題が発生する．

ビッグデータの代表として取り上げられるデータは，クリックストリームデータである．このデータは，イベント形式時系列データである．クリックストリームデータは，ブラウザー上で，クリックをした時点で，イベントとしてサーバーに送り込まれ，蓄積されたデータである．このようなイベント形式時系列のデータは，本来の業務中に発生するデータの蓄積であり，いたるところで発生している．

このようなイベント型の時系列データは，状態遷移によるモデル化を行い，状態遷移の状況を把握・観察することにより，データ分析を行うことが可能である．イベントにより状態変化が伴う形式の状況は多い．携

まつもと　しんや
日本テラデータ（株）
〒 107–0052 東京都港区赤坂 2–23–1 アークヒルズフロントタワー

帯電話の契約に関して，契約と解約というイベントが契約状態と非契約状態の間の状態遷移を示す．これは，状態とイベントが明確に結びついた場合である．逆に，明確な状態遷移が既知でない場合には，どのようなイベントが状態遷移に結びついているかを見つけ出し，状態間の推移を分析することにより，知見を得ることができる．

3. 分析環境

弊社では，大規模なデータを取り扱うことが可能なデータウェアハウス用のRDBMSを販売してきた．すでに，数十ペタバイトの容量を持つシステムが稼働している．理論的に可能であるとか，実験システムで稼働したというのではなく，本番のシステムとして稼働している．言い換えると，ビッグデータを特徴づける三つの項目のうち，量と速度の項目を満たすRDBMSを何年も前から販売・稼働させている．このため，弊社では，量と速度の項目が，ビッグデータの範囲に到達しているだけで，RDBMSで取り扱えるシンプルな形式のデータを，ビッグデータとは呼んでいない．

昨年，弊社はAsterData社を買収し，Teradata Asterというビッグデータ解析用プラットフォームの販売を開始した．Teradata Asterは，MapReduce技術 [3] を用いて複数のコンピュータを接続した分散コンピューティング環境である．そして，ビッグデータの分析を簡単な命令で実行できるように，RDBMSの標準言語であるSQLを拡張した [4]．これは，SQL/MapReduce技術と呼ばれ，米国において特許化された．

さらに，単なるSQLの拡張だけではなく，高度な分析も簡易な記述により分析が可能なように，いくつかの分析アルゴリズムを用意している．特に，XMLなどで記述された情報を，元の形式で蓄積しておき，分析時点で読み解釈をすることで，状態を把握する機能を用意しており，ほかのビッグデータ解析用プラットフォームとの差別化を図っている．

4. 米国事例

4.1 分析対象業務

米国における事例として，オンラインゲーム会社の事例を取り上げる．

ゲーム会社は，お金をいただいて楽しんでもらうことをビジネスとしている．このためには，ゲームの分野などから限界があるものの，間口を広げ，多くのプレイヤーに参加してもらうことが必要である．また，オンラインゲームでは，長くゲームを続けてもらうことも必要であり，新規の要素の追加など飽きさせないための工夫が必要である．

日本では，携帯電話を用いたオンラインゲームが普及しているが，本稿では，パソコンなどで行う多人数参加型のオンラインゲームを対象とする．このオンラインゲームでは，プレイヤーは戦場を駆け巡り，敵を倒す．このようなゲームでは，ゲームの発売前に一般のユーザーを交えて，ベータテストというものを行う．これは，プレイヤーが想定外の行動をした際に，侵入不可能な領域に侵入したりなど，ゲーム中で想定外のおかしな動作をしないか確認する目的もあるが，ゲームを楽しめるものとするために，細かな調整を行うことも目的としている．そして，ベータテストで一般ユーザーに使ってもらった以上，早期に市場に投入する必要があり，ベータテストの結果分析に時間をかけることはできない．以前は，テスト開始前から分析項目を絞り込み，データの採取項目を絞り込んでいた．最近は，「ビッグデータ」という言葉で表される大量データ分析の方法が発達し，分析項目を増やすことが可能となった．そして，当初に想定した分析から派生した分析を実施することで，より深い分析が可能になった．

ビッグデータとして深い分析をすることにより，以前から行ってきた不具合修正やバランス調整だけではなく，ゲーム中で不公正な行動を見つけ出すことが可能になった．不公正な行動とは，本来のゲームで想定した範囲を超えて，一方に有利な行動に対する総称である．例えば，ある位置で待っていることで容易に敵を倒すことができる場合がある．これが，現実でも可能なことであれば，仕様と考えられるが，コンピュータゲーム特有で，有利になる行動は，不公正な行動とみなせる．また，親しいプレイヤーが敵味方に分かれて，ほかのプレイヤーの行動を教えたりしてねらうことなどは，現実の戦闘として可能なことであるが，裏切り行為である．これも不公正な行動と考えられる．このような不公正な行動を見つけ出し，このような行動を起こしにくくなるような調整を行うことが可能となった．

このような多人数参加型のオンラインゲームでは，1秒間に数回程度，サーバーとユーザーPCとの間で，データがやりとりされる．ユーザーがどのような行動を行ったのかが，ユーザーPCからサーバーへ送信される．逆に，他ユーザーの行動・位置情報とともに，サーバーが自動的に操作しているキャラクターの行動が，サーバーからユーザーPCに送信される．これら

表 1　イベントデータ例

roundid	playerid	roundtime	message
1	1	2011/11/12 10:10:01	/locationx=1239 /locationy=1223 /action=walk /state=alive
1	1	2011/11/12 10:10:02	/locationx=1239 /locationy=1224 /action=walk /state=alive
1	1	2011/11/12 10:10:03	/locationx=1239 /locationy=1225 /action=walk /state=alive
1	1	2011/11/12 10:10:04	/locationx=1239 /locationy=1226 /action=walk /state=alive
1	1	2011/11/12 10:10:05	/locationx=1239 /locationy=1227 /action=walk /state=alive
1	1	2011/11/12 10:10:06	/locationx=1239 /locationy=1228 /action=walk /state=alive
1	1	2011/11/12 10:10:07	/locationx=1239 /locationy=1229 /action=walk /state=alive
1	1	2011/11/12 10:10:08	/locationx=1239 /locationy=1229 /action=fire /state=alive /targetx=1259 /targety=1224
1	1	2011/11/12 10:10:09	/locationx=1239 /locationy=1229 /action=fire /state=alive /targetx=1259 /targety=1224
1	1	2011/11/12 10:10:10	/locationx=1239 /locationy=1229 /action=none /state=dead
1	2	2011/11/12 10:10:01	/locationx=1239 /locationy=1223 /action=walk /state=alive
1	2	2011/11/12 10:10:02	/locationx=1239 /locationy=1224 /action=walk /state=alive

```
SELECT …
FROM nPath (
  ON …
  PARTITION BY roundid, playerid
  ORDER BY eventid
  …
  PATTERN('ALIVE.DEAD')
  SYMBOLS (
    event = 'PlayerStart' OR event = 'Spawn' AS ALIVE,
    event = 'Death' OR event = 'PlayerEnd' AS DEAD
  )
  RESULT (
    FIRST(playerid OF ALIVE) AS playerid,
    FIRST(roundid OF ALIVE) AS roundid,
    FIRST(roundtime OF ALIVE) AS start_time,
    FIRST(roundtime OF DEAD) AS end_time,
    FIRST(durationseconds OF ALIVE) AS durationseconds,
    FIRST(roundstartdate OF ALIVE) AS roundstartdate
  )
) n
```

図 1　生存時間解析用プログラム

の交換されるデータは，ゲームの状況を再現するための最低限の情報に絞り込まれている．そして，これらの情報はオンラインゲームの機能追加に柔軟に対処するために，XML 形式でやりとりされている．今回紹介する事例では，ユーザー PC から送られた情報をすべて蓄積し，ビッグデータとして分析した．

4.2　分析 1 ：各ラウンドでの生存時間の分布

本稿で扱うゲームは，「ラウンド」という単位で 1 回のゲームが成り立っている．プレイヤーがラウンド開始または途中参加してゲームを開始してから，ラウンド終了，または戦闘不能になるまでの時間を生存時間としている．この生存時間は，ゲームに参加している時間であり，ゲームを楽しめている時間として重要な指標である．また，各プレイヤーが不正を行っていないかを見つけ出すために，生存時間が適切に分布しているかを確認することは重要である．生存時間は，連続して分布していると考えられる．しかし，特殊な行動を起こすことにより，生存時間が延びることも考えられる．これは，特殊な行動が生存時間を延ばすことにつながり，不公平な行動となるので，ゲームとして避けなければならない．そのような状況が発生していないか，生存時間が連続的に分布しているかを確認する．

表 1 に通信データの例を示す．これは，位置情報とともに，その時に発生したこと (EVENT) や状態 (STATE) の送信データである．図 1 に生存時間解析用のプログラムを示す．Teradata Aster の nPath 関数は，SQL における FROM 句の一部のように記述する．各イベントのフィールド内の項目を評価して，現在どのような状態であるかを決定する．「Message」フィールドの「event」が，「PlayerStart」か「Spawn」であれば，「生存中」という状態を開始するイベントである．「Death」か「PlayerEnd」であれば，「生存中」が終了したイベントであり，これ以降を「死亡状態」であるとみなす．ここでは，「生存中」と呼んでいるが，プレイヤーがゲームを開始したというイベントである．また，「死亡状態」というのは，ゲームオーバーとステージクリアの両方の状態を含んで，ゲームを終了したことを示す．この「生存中」の最初の時刻を start_time として取り出す．また，「死亡状態」の最初の時刻を end_time として取り出すことで，ゲーム時間（生存時間）を計算することができる．これらの生存時間を元に，ほかの分析を実施する．

4.3 分析２：プレイヤー関連性分析

複数のプレイヤーが同時に参加するゲームであり，各ラウンドに同時に参加しているプレイヤーは，親密度が高いと考えられる．

これについて，Teradata Aster の関連性分析機能を用いて分析を実施した．関連性分析はバスケット分析という名前でも知られており，紙おむつと缶ビールが同じバスケット入ることが多いことを見つけ出した方法である [5]．何をバスケットとしてみなすか，何を商品としてみなすかにより，応用範囲の広い手法である．筆者は，関連性分析の医療分野への応用を行っている [6]．本分析では，ラウンドを「バスケット」としてみなし，プレイヤーを「バスケット」に入る「商品」とみなして分析を実施した．関連性分析では，ありふれた組合せが大量に出力されるので，注意深く読み解くことが必要である．例えば，「寿司」と「緑茶」は一緒に買われることが多い．このようなありふれた組合せがたくさんみつかる．ありふれていない組合せにたどり着くために，他の情報と突合せを行う必要がある．

図 2 に，関連性分析の結果例を示す．同じゲームに参加している回数が多いユーザーをつなぎ合わせることで，プレイヤー同士のコミュニティを見つけ出すことができる．

親密度が高いだけでは，大きな意味を持たない．複数のプレイヤーが集まることによって可能となる不公正な行動も存在する．親密度が高い場合は，複数プレイヤーによって可能となる不公正な行動を行っていないかの監視が必要であり，他の分析結果と組み合わせて不公正な行動を行っているかを判断する．

この分析は，ソーシャルネットワークサービス（SNS）をはじめとする社会ネットワークにおけるコミュニティの発見とそのコミュニティにおけるインフルエンサーの発見を行うことに応用可能である．

4.4 分析３：接近レポート

プレイヤー同士はチームを組んで，ゲームの中で戦闘を行う．逆に，チーム以外のプレイヤーは，近づくことはほとんどない．チーム以外のメンバーが接近することで，通常では得られないような得点を得ていることも考えられる．例えば，知り合いが敵味方に分かれ，お互いの動きをみること，つまり敵の動きを知ることで有利な行動が可能になる．また，わざと自分を殺させることで，敵となった知り合いに得点をあげさせることができる．ゲーム上可能であっても，ゲームが再現したい現実とは異なり，不公正な行動である．これは，他のプレイヤーのやる気を殺ぐことにもつながり，

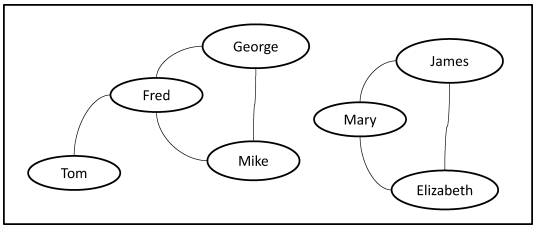

図 2 関連性分析の結果例

```
SELECT …
FROM
(
  SELECT…
  FROM recv_log_pc_32
  WHERE event = 'Kill'
) t1,
npath (
  on (
    SELECT…
    FROM recv_log_pc_32
    WHERE event in ('Spawn', 'Death')
  )
  PARTITION BY roundid, playerid
  ORDER BY eventid
  MODE(NONOVERLAPPING)
  PATTERN('SPAWN.DEATH')
  SYMBOLS (
    event = 'Spawn' AS SPAWN,
    event = 'Death' AS DEATH
  )
  RESULT(
    …
  )
) t2
WHERE t1.roundid = t2.roundid
AND t1.eventid = (t2.eventid - 1)
AND t2.alive_time < 10;
```

図 3 登場直後死亡プログラム

排除すべき行為である．プレイヤーのイベント情報から，プレイヤー同士の距離を計算し，その距離がある閾値以下の場合にレポートとして出力する．この分析を実施する機能は用意されていなかったので，Java 開発機能を用いて関数を作成した．

4.5 分析４：ゲーム登場直後の死亡

ラウンドの途中から参加するプレイヤーが，登場直後に，戦闘不能とさせられることがある．ラウンドの開始から参加しているプレイヤーは，そのラウンドの状況をラウンド開始前にサーバーから送信される．しかし，ラウンド途中から参加する場合には，その時点までにほかのプレイヤーが行動を行った結果も含めた状況を，サーバーから取得する．このようなタイムラグが発生することを含めて，ゲーム登場直後は，周囲

表 2　登場直後の死亡検索結果例

Roundid	Killer_playername	Alive_time	Weaponname	Positionx	Positiony
1	Jumbo100	5.4	AK74	389	787
1	Jones99	4.1	M16A4	231	375
2	Jumbo100	5.4	AK74	389	787
2	Patton43	4.3	M16A4	983	452
2	Jumbo100	6.2	AK74	389	792
2	Tom102	4.5	RPG-7	278	132
2	George87	4.5	RPG-7	983	121
3	Jumbo100	5.4	AK74	389	787
3	Kate09	5.6	AK74	392	794
3	Jumbo100	5.4	AK74	389	782
3	George87	5.3	AK74	391	787

の状況が把握できない．このため，プレイヤーの行動は制限される．ゲームシステムの制約により行動が制限されているなかで，死亡することは，ゲームの興味を喚起する部分まで到達できなくなる．プレイヤーの興味を失わせ，解約につながる．また，途中から参加するプレイヤーをねらいうちすることで，高得点をあげることができるということは，不公正なゲームである．このような状況を解消する必要がある．

分析としては，ゲームに登場してから死亡するまでの時間が極端に短いプレイヤーを特定し，そのプレイヤーがだれによって殺されたのか，どこで殺されているのかを見つけ出す．図 3 に，登場直後の死亡の分析用プログラムを示す．nPath 関数により，登場してから死亡するまでの時間がどの程度であるかを求めている．これは，図 1 に示したラウンド中の生存時間と同じものである．そして，通常の SQL と同様に，「Death」イベントが発生した直前のイベントを，eventid が一つ前であることで特定し，「Kill」イベントを特定する．

表 2 に，登場直後死亡の分析の検索結果例を示す．どのラウンドで，だれが，どの武器を用いて，どこで Kill したのかが検索される．特定のプレイヤーや特定の場所で殺されている場合には，ゲーム仕様として問題がある可能性がある．つまり，登場直後で Kill されやすい場所が存在し，特定のプレイヤーがその事実を知っていることが想定される．詳細な状況確認を行い，このような場所を解消する必要がある．

4.6　分析 5：マップ・グリッド移動

プレイヤーやチームがゲームマップ上のどのグリッドを移動したのかを分析する．これは，ゲームの拡張部分の開発を計画する材料となる．オンラインゲームは，プレイヤーを飽きさせないために，継続的に開発を行い，拡張パックとして提供を行っている．プレイヤーの移動パターンが固定化された場合には，マンネリ化してきており，飽きてきている可能性が考えられる．ゲームの拡張を実施する必要がある．

この分析は，次の手順で実施した．

(1)　位置情報をグリッド情報に変換する
(2)　グリッド情報を，状態とみなし，nPath 関数を用いて，状態変化のリストとして取り出す．
(3)　リストとして取り出した移動経路の順序ごとに件数をカウントする．

このような単純な分析手順である．しかし，ビッグデータ分析で重要なことの一つに，データのコピーを作成しないということがある．コピーを作成することにより，コンピュータ資源がひっ迫につながるためである．並列で処理可能な個所は並列で処理を行い，同一の CPU 上で引継ぎか可能ならば，結果のコピーを作成せずに次の分析に移り，中間結果を同一の 1 次キャッシュないし主記憶内に作成し，その場で捨てていくということが可能になる．この分析では，位置情報からグリッド情報に変換し，その移動経路を作成するまでは，各プレイヤー，各ラウンドに閉じており，その中で，一連の処理を行う．そして，移動経路ごとの集計は，全体を対象とする必要があるので，全体で実行する．

各処理が並列に処理されていることを意識することにより，適切な分析処理を記述することが可能である．しかし，SQL/MapReduce の環境では，細かくどのような並列処理がなされているかを意識する必要はなく，どのような単位で並列処理がなされているかを意識するだけで十分であり，本来の分析方法を考えることに集中しやすい．

5.　おわりに

本稿は，オンラインゲームのベータテストにおける

通信データを元に複数の分析を実施した事例を紹介した．これらの分析は，実施前から分析手法を綿密に計画することにより，以前から可能であった．しかしながら，分析を行うためには分析方法を絞り込み，蓄積する項目を絞り込んでおく必要があった．本事例では，生のログデータを蓄積し，必要な項目を分析中に取り出すことで，分析を行った．一つの分析だけを行う場合には，不要なデータを読むことにもなるが，1 回の読み取り (1 pass read) とすることで，トータル分析時間を短くできる．

分析用の SQL 開発時間は，Teradata Aster で用意された関数を用いた場合では，拡張 SQL 文を 10 分程度で記述できる．Java 言語を用いて，新たに関数を作成する場合では，2 時間程度の時間が必要であった．1 回の分析処理で結論を導き出せないような分析を行うためには，一つの分析結果を受けて，思考を止めることなく次の分析を行うほうがよい．つまり対話的な分析がよい．対話的な分析を支援するための分析環境として，ハードウェア的な側面だけではなく，分析を簡単に記述し実行する分析用プラットフォームが必要である．

参考文献

[1] H. Akaike, A new look at the statistical model identification. *IEEE Trans. Automatic Control*, **19**, 716–723, 1974.

[2] G. Kitagawa and W. Gersch, A smoothness priors-state space modeling of time series with trend and seasonality. *Journal of the American Statistical Association*, **79**, 378–389, 1984.

[3] J. Dean and S. Ghemawat, MapReduce: Simplified data processing on large clusters. Proceedings of the Sixth Symposium on Operating System Design and Implementation (San Francisco, CA, Dec. 6.8). Usenix Association, 2004. (http://labs.google.com/papers/mapreduce.html)

[4] E. Friedman P. Pawlowski and J. Cieslewicz, SQL/MapReduce: A practical approach to self-describing, polymorphic, and parallelizable user-defined functions. *Proceedings of the VLDB Endowment*, **2**, 1402–1413, 2009.

[5] R. Agrawal, T. Imielinski and A. Swami, Mining association rules between sets of items in large databases. *Proceedings of ACM SIGMOD International Conference on Management of Data*, 207–216, 1993.

[6] T. Imamura, S. Matsumoto, Y. Kanagawa, B. Tajima, S. Matsuya, M. Furue and H. Oyama, A Technique for identifying three diagnostic findings using association analysis. *Medical and Biological Engineering and Computing*, **45**, 51–59, 2007.

