値域分割に基づく分散ストレージにおける効率向上のための複製管理

A Replica Management for Range-Partitioning based Distributed Storage to Improve Efficiency

渡邊 明嗣♡
横田 治夫♠

Akitsugu WATANABE
Haruo YOKOTA

値域分割に基づく分散ストレージの効率向上には複製の効果的な管理が不可欠である．本稿では，値域分割ベースの分散ディレクトリとプライマリ・バックアップ型のデータ冗長化を組み合わせた分散ストレージシステムにおける，アクセス負荷分散と容量分散を両立する複製配置の管理手法を提案する．提案手法は従来の負荷計測ならびにそれを前提とした配置戦略との親和性が高く，広い応用範囲が見込まれるものである．

To improve performance of distributed storage, it is necessary for effective management of relicas of each data item. In this paper, we propose a replica management method balancing both access loads and relative space utilization simaltanously. In a primary-backup based distributed storage systems, the proposed method has wide applicability combined with comventional load evaluations and data migration methods.

1. 序論

現在，ストレージシステムに求められる可用性と容量利用効率は益々高まってきている．また，近年のストレージに格納されるデータ量の増加を鑑みるに，より単純な手法，すなわち単純さから来る堅牢さを持ち合わせた手法によって可用性と容量利用効率を高めることの意義はより大きなものとなっている．

これらの要求に対する解法として，複数のディスクを用いた分散ストレージ構成のうち，データを冗長化し，データセットを分割して複数のディスクへ配置する手法を改善することが可用性と規模拡張性向上の要となっている．データの冗長化手法としては，データオブジェクトをそのまま複製する手法，パリティなど符号処理を用いて冗長なビットを付与する手法などが挙げられるが，データを複製する手法は，冗長なビットを用いる手法より単純で堅牢であるのみならず，データ更新および故障後の復帰処理のコストが低いという長所を持っている．

論理的なデータ領域を物理的な複数のディスクに割り当てるに当たって，一般に，多くのアプリケーションはデータを水平分割する．水平分割戦略は，ラウンドロビン，ハッシュ，値域分割の 3 つに大別される [1].

いずれも各々長所と短所を併せ持っている．値域分割はレンジクエリと完全一致クエリの処理に優れている．また，連続データを物理的に隣接したディスクページに配置するため，多くのアプリケーションに対してより良いアクセス性能を提供できる．しかしながら，値域分割には偏りの問題がある．ラウンドロビンは偏りが無い代わりに，規模拡張性に問題がある．ハッシュは完全一致クエリの処理に優れ，偏りも小さいが，データ配置をランダム化するため，レンジクエリや連続データの取り扱いに問題がある．また，ハッシュ分割では，データ再配置に際して全データのハッシュ値を再計算する必要があるため，システム再構成を考慮した場合の規模拡張性は制限される．一方，値域分割は偏りが生じるという短所を持つが，これはディスク間でデータを移動させることで解消できる．このような動的データ再配置は，アクセスパターンの変化やシステム構成の変化による偏りに対しても効果を発揮する．

アクセス性能の向上と値域分割に対する効率的な動的データ再配置を行うためには，適切な分散ディレクトリが必要である．Fat-Btree は，分散更新とデータ再配置を考慮した値域分割ベースの分散ディレクトリ構造である [2]．しかし，横田が当初提案した偏り除去アルゴリズムはデータ量の偏りにのみ注目したものであり，アクセス負荷偏りに対して効果的では無かった．他方，アクセス負荷偏りに対する偏り除去手法には，データ量偏りを考慮していないという問題があった [3, 4, 5].

本稿の目的は，アクセス負荷とデータ量の偏りを同時に解消することを目的とした複製管理戦略の提案である．提案手法は複製を用いた手法の特長である高いサービス品質と，値域分割の特長であるレンジクエリやアクセス性能における利点を併せ持ち，また，非同期更新を用いることで更新操作のコストをも抑えている．これらの特長により，提案手法は分散ストレージシステムにアクセス性能と容量の利用効率の両面において高い規模拡張性を提供しうる．

2. 章では非同期バックアップを用いた場合のアクセス負荷分散について述べる．3. 章では，前章の議論を発展させ，新しい配置手法の提案とその特徴について述べる．4. 章では本研究の関連研究について述べる．5. 章では本稿の結論と，将来の課題について述べる．

2. アクセス負荷の分散

アクセス性能と容量の利用効率の両面において高い拡張性を実現するためには，アクセス負荷とデータ量の偏りを共に解消することが有効である．本節では，アクセス負荷と容量利用率の同時均等分配について検討する前に，値域分割のみを用いた場合，および，複製を用いたデータ冗長化である chained declustering[6] のみを用いた場合の負荷分散手法を検討する．なお，以下の議論では簡単化のため，分散ストレージを構成する各ユニットが均質である場合のみを考える．

2.1 値域分割とアクセス負荷分散

各ディスクの容量利用率が釣り合っている場合でも，偏ったアクセス分布などの要因によってアクセス負荷は偏りうる．データを高負荷状態のディスクから低負荷状態のディスクに移動させることによってアクセス負荷の釣り合いを取ることができる．値域分割では，データの並び順を保持する必要があるため，データ移動は隣接するディスク間でのみ行われる．

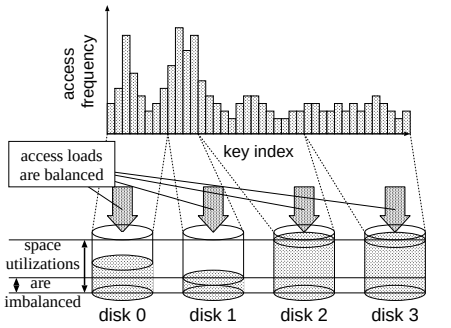
図 1 は，アクセス負荷の釣り合いを取るためのデータ移動を適用した結果，容量利用率の釣り合いが崩れた状態を示している．図の上部はアクセス頻度，下部は容量利用率を表している．中央の点線は値域とディスクとの間の対応を示している．最下部に示した数字は容量利用率を%表記で示したものである．このように，アクセス負荷の釣り合いを取るためのデータ移動は容量利用率の釣り合いを崩す可能性がある．

2.2 Chained declustering とアクセス負荷分散

次に，システムの可用性について検討する．高いアクセス性能と単一ディスク故障への耐性を両立させるという目的に対しては，chained declustering ベースの複製を用いた障害対策と値域分割を組み合わせる手法が有効である [7].

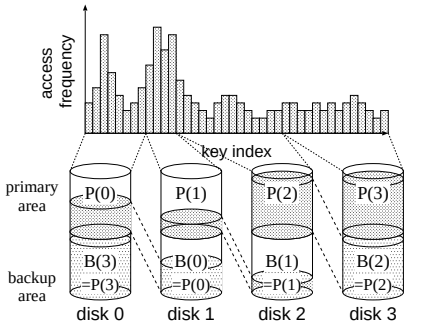
本稿では，議論を簡単にするため複製数を 2 に限定する．以下の議論では，先行書き込みログを利用した非同期更新が行われることを仮定する．非同期更新では，まず操作のログをメモリ上に格納しプライマリ複製を更新した時点で操作の完了通知を発行する．しかる後に，アイドル時間を用いて他の複製を更新する．このような非同期更新では，バックアップコピーがいわゆるダーティーコ

♡ 学生会員　東京工業大学 大学院 情報理工学研究科 計算工学専攻
aki@de.cs.titech.ac.jp
♠ 正会員　東京工業大学 学術国際情報センター　yokota@cs.titech.ac.jp



	disk 0	disk 1	disk 2	disk 3
P	24%	12%	42%	42%
total	24%	12%	42%	42%

図 1: アクセス負荷均衡化によって生じた容量利用率偏り
Fig.1 Data skews to balance access load for a skewed access pattern



	disk 0	disk 1	disk 2	disk 3
P	24%	12%	42%	42%
B	42%	24%	12%	42%
total	66%	36%	54%	84%

図 2: chained declustering におけるデータ偏り
Fig.2 Data skews under chained declustering

ピーとなりうるため，アプリケーションの目的によって各複製へのアクセス頻度が変化しうる．この事を定式化して表すため，本稿ではプライマリコピーから読み出される割合を ψ ($0 < \psi \leq 1$) とおき，アプリケーションのアクセスに偏りがあり，かつ，ダーティーコピーの利用頻度が低い ($\psi \simeq 1$) 場合のみを検討する．

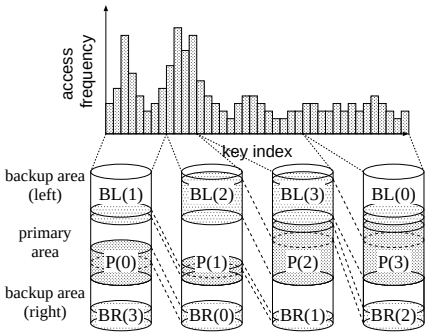
図 2 は，非同期更新を前提とした chained declustering ベースのデータ配置の例である．n 番目のバックアップ領域 B(n) は隣接ディスクに格納された n 番目のプライマリ領域 P(n) の複製である．P(n) は n 番目のディスクに格納され，B(n) は $n+1 \bmod N$ 番目のディスクに格納される．ただし，N はディスクの総数である．P(n) と B(n) は異なるディスクに格納されるため，いずれかのディスクが故障した場合でもデータは失われず，サービスを継続することができる．この配置法では隣接する 2 つのディスクが同時に故障しない限り，データ破損が生じない．

データ偏りの観点から見ると，たしかに chained declustering にはデータ偏りを減らす効果がある．しかし，複製なしの場合 (図 1) と chained declustering(図 2) における，同じアクセスパターンにおける容量利用率の違いを示した例にも表れているように，その効果は十分ではない．

3. 適合的複製分割管理手法

アクセスが偏っている場合の容量利用率を向上させるため，本稿では複製されたデータを分割し，適合的に再配置するデータ配置手法を提案する．提案手法は chained declustering を発展させたもので，データ量の偏りを減らし，バックアップコピーの再生成時間を減らすことで可用性をも高める．

提案手法では，バックアップ領域を 2 つに分割し，プライマリコピーを格納するディスクに隣接する 2 つのディスクに各 1 つずつを格納する．しかる後に，分割されたバックアップ領域間の境界を調整することで，この 2 つのディスク間のデータ量を操作し，容量利用率を均衡化させる．この分割はアクセス偏りおよび容量利用率を均衡させやすくし，しかもバックアップコピーの再生成時間をも減らすものである．



	disk 0	disk 1	disk 2	disk 3
BL	6%	21%	21%	12%
P	24%	12%	42%	42%
BR	21%	12%	6%	21%
total	51%	45%	69%	75%

図 3: 適合的バックアップ分割によって均衡化された容量利用率とアクセス負荷
Fig.3 Data and access load balance with adaptive backup division

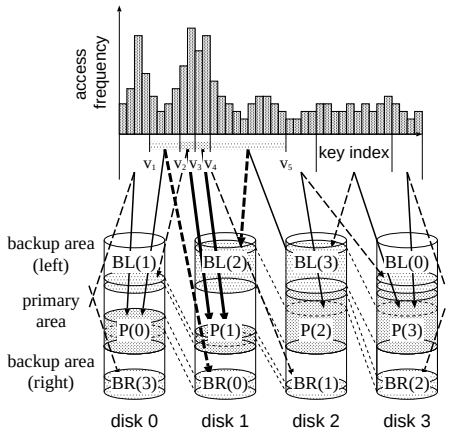


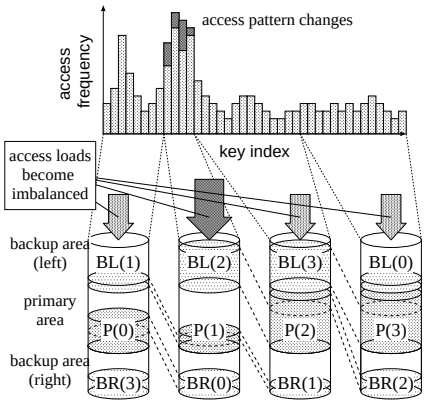
図 4: ディスクに割り当てられた値域とその重なり
Fig.4 An overlapped continuous range on a disk

3.1 適合的バックアップ分割

多くのアプリケーションではダーティーコピーを利用しない ($\psi \simeq 1$)．したがって負荷分散の観点から見ると，バックアップの配置はプライマリの配置ほど重要では無い．バックアップをプライマリとは独立に配置することが可能であれば，プライマリの配置を変えることでアクセス負荷を分散し，バックアップの配置を変えることでデータを分散することができるようになるだろう．chained declustering [6] ではプライマリ領域 P(n) のバックアップ B(n) を片方の隣接ディスクにのみ格納していたが，提案手法では，これを BR(n) と BL(n) の 2 つに分割して，それぞれ右と左の各隣接ディスクに格納し，容量利用率の偏りを減らす (図 3)．分割条件 C_a は以下の通りである：

1. P(n) ≡ BR(n) ∪ BL(n)
2. BR(n) ∩ BL(n) ≡ ∅
3. BR(n) と BL(n) は $P(i)$ をキー $k(i)$ で分割したもの

この分割によって，2 つのディスク上の領域 BR(n) と BL(n) との間でデータを融通できるようになる．すなわち，アクセス負荷に対する配置とは別個にデータ量に対する配置を行うことができるようになる．



	disk 0	disk 1	disk 2	disk 3
BL	6%	30%	12%	6%
P	24%	12%	42%	42%
BR	30%	18%	6%	12%
total	60%	60%	60%	60%

図 5: 偏り除去前のアクセス負荷が偏っている状態
Fig.5 Load transitions before balancing

図 3 は，図 2 と同じ条件のものに対してこのバックアップ分割を適用した例である．この例では，他の手法を用いた場合と同様にアクセス負荷を平均化しつつ，容量利用率も完全に平均化されている．

適合的バックアップ分割は容量利用率の完全な平均化を保証しない．しかし，様々なアクセス分布に対して，この手法は大いに有効であると期待できる．

ある 1 つのディスクに格納されているデータアイテムの値域に注目すると，両脇を 2 つのバックアップ領域で挟まれたプライマリ領域という連続した値域が各ディスクに割り当てられることになる（図 4）．これらの連続した値域はディスク間で重なり合っている．そのため，アクセス負荷均衡化のためのデータ移動操作を，対応する 1 対の複製を同一ディスク内のプライマリあるいはバックアップ領域に割り当てなおすという一連の低コストの操作群で代用できる．

この手法は Teradata の interleaved declustering [8] とバックアップを分割するという点で類似しているが，分割されたバックアップの配置方法に大きな違いがあり，そのため，得られるデータの生存性の規模拡張性が大きく異なる．Interleaved declustering は連続する 2 つのディスク故障に対して無防備だが [6]，この手法では chained declustering 同様，隣接する 2 つのディスク故障が起きない限り，データは消失しない．

3.2 データ移動アルゴリズム

データ量およびアクセスパターン変化時にはデータ移動が必要となる．ここでは，容量利用率均衡化のためのディスク間データ移動アルゴリズムとディスク内データ移動アルゴリズム，および，アクセス負荷均衡化のためのアルゴリズムについて述べる．無論，これらのアルゴリズムは併用できる．

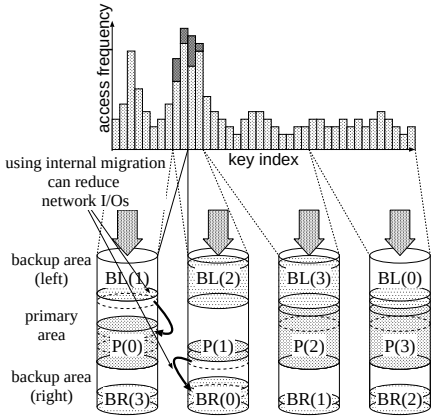
3.2.1 負荷移動のためのディスク間データ移動

システムには，データ量およびアクセス負荷の偏りを検出する何らかの機構（例：TCSH[9]）が既に備わっていると仮定し，本稿ではその詳細については触れない．アクセス負荷偏り検出機構が出力する最も負荷が高いディスクを (h)，期待されるデータの移動先を (d)，必要なデータ移動量を (m) とおく．ただし，配置ルールの制約を考慮した移動先が選ばれるものとする．

ディスク間データ移動は chained declustering におけるデータ移動と同様のアルゴリズムによって処理される．次節で述べるディスク内データ移動の場合と同様に，移動元のバックアップの量が十分であれば各ディスクの容量利用率は変化しない．

```
Intra_Disk_Migrate (h, m, d)
begin
  if d= h− 1 mod N then
    Let i= min (m, cardinality(BL(h));
    Migrate idata items from P(h) to BR(d);
    Migrate idata items from BL(h) to P(d);
    if (i< m) then
      Migrate (m− i) data items from P(h) to P(d);
      Migrate (m− i) data items from BR(h) to BR(d);
  else
    Let i= min (m, cardinality(BR(h));
    Migrate idata items from P(h) to BL(d);
    Migrate idata items from BR(h) to P(d);
    if (i< m) then
      Migrate (m− i) data items from P(h) to P(d);
      Migrate (m− i) data items from BL(h) to BL(d);
end
```

図 6: ディスク内データ移動アルゴリズム
Fig.6 Intra-disk data migration algorithm



	disk 0	disk 1	disk 2	disk 3
BL	0%	30%	12%	6%
P	30%	6%	42%	42%
BR	30%	24%	6%	12%
total	60%	60%	60%	60%

図 7: ディスク内データ移動によるアクセス負荷分散
Fig.7 Load balancing by intra-migration

3.2.2 負荷移動のためのディスク内データ移動

ここでは，3.1 節で述べたアクセス負荷均衡化のためのデータ移動操作を，複製間の役割変更を利用することによって，より低いコストで行うアルゴリズムについて述べる．

図 3 の状態から図 5 の状態へとアクセスパターンが変化した場合を例に，図 6 に示したディスク内データ移動アルゴリズムの動作を説明する．P(1) にあるデータアイテムは同じディスク 1 上に格納された BR(1) に再割り当てされ，同時に BL(1) にあるデータアイテムは同じディスク 0 上に格納された P(0) に再割り当てされる．

BL(1) に十分な量のデータアイテムがあるならば，これらの操作はネットワーク帯域を一切消費しない．しかも，これらの操作はデータアイテムのメタデータを書き換えるだけで実現可能なので，実際のデータ I/O そのものも減らすことができる．図 7 が示すように，この操作は各ディスクのデータ量を変化させない．BL(1) に十分な量のデータが無かった場合は，BL(1) から BR(0) へデータアイテムを移動した後に，BR(1) から BR(0) へデータアイテムを移動させる．この操作はこの 2 つのディスクのデータ量を変化させる．

3.2.3 データ量移動のためのディスク間データ移動

提案手法はデータ挿入や削除などによって容量利用率のバランスが変化した場合にも有効に働く．このような場合には，アクセ

```
Balancing_Data (h, m)
begin
  Let ls be the list of data amount on each disk;
  Let i= m;
  Let a= (ls[h] + ls[h−2 mod N] + ls[h+2 mod N])/3 ;
  Migrate min(cardinality(BL(h+1 mod N)), a-ls[h−2 mod N], i)
    data items from BL(h+1 mod N) to BR(h+1 mod N);
  Subtract the number of migrated data items from i;
  Migrate min(cardinality(BR(h−1 mod N)), a−ls[h+2 mod N], i)
    data items from BR(h−1 mod N) to BL(h−1 mod N);
  Subtract the number of migrated data items from i;
end
```

図 8: データ量移動のためのディスク間データ移動アルゴリズム
Fig.8 Data volume balancing algorithm

ス負荷の均衡を破らないように，プライマリデータセットの配置を維持したまま，その複製であるバックアップデータセットのみを移動させて容量利用率を調節する．図 8 はこの容量利用率調整アルゴリズムの概要である．

4. 関連研究

障害からの回復を速めるため，複製を用いた戦略が研究されてきた．Gamma [10] で用いられている chained declustering [6] については前述の通りである．Teradata の interleaved declustering は，ディスク故障からすぐにデータ量とアクセス負荷のバランスをとることができる [8] が，データの生存性に規模拡張性が無いため大きなシステムでは対故障性能を期待できず，また，ハッシュを用いた分割しか利用できないという欠点がある．

動的偏り制御の分野では様々な研究が行われている．Scheuermann らは効果的な偏り除去においてはデータ移動のコストそのものも考慮すべきであると述べている [3]．また，Btree ベースのディレクトリを備えたシステムでは部分木単位での BULK ページ移動による高速なデータ移動手法が可能であることが知られている [4]．RING 手法では，Btree ベースのディレクトリの両端をつなげることで，アクセス負荷均衡に必要なデータ移動量を減らしている [11]．

5. 結論

本稿では，値域分割に基づく分散ストレージの効率する複製管理戦略の一端として，アクセス負荷とデータ量の偏りを同時に解消するデータ配置戦略を提案した．提案手法では，プライマリ領域を格納したディスクに隣接した 2 つのディスクに適合的バックアップ分割手法によって分割されたバックアップ領域を格納する．その結果，各ディスクは 2 つのバックアップ領域と 1 つのプライマリ領域からなる連続した値域を割り当てられることになる．2 つのバックアップ領域の大きさを調整する適合的バックアップ分割は，アクセス負荷を均衡させつつ領域利用率の偏りを従来手法よりも小さく抑えるという目的に対して効果的である．負荷移動のためのディスク間／ディスク内データ移動とデータ移動のためのディスク間データ移動を組み合わせることで，アクセス分布やデータセットが変化した場合でも，アクセス負荷とデータ量の偏りを同時に解消することができる．

今後の展望としては，配置戦略をより洗練させ多数の複製に対応し，また，Fat-Btree の特性に特化した仕組みを組み上げることでさらなる性能の向上を図り，さらには，シミュレーションならびに実機実験を行って従来手法との比較を行う予定である．

[謝辞]

また本研究の一部は，独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST，情報ストレージ研究推進機構 (SRC)，文部科学省科学研究費補助金特定領域研究 (16016232)，および東京工業大学 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行なわれた．

[文献]

[1] David DeWitt and Jim Gray. Parallel Database Systems: The Future of High Performance Database Systems. *Communications of the ACM*, Vol. 35, No. 6, pp. 85–98, June 1992.

[2] Haruo Yokota, Yasuhiko Kanemasa, and Jun Miyazaki. Fat-Btree: An Update-Conscious Parallel Directory Structure. In *Proc. of 15th Int'l Conf. on Data Engineering*, pp. 448–457, 1999.

[3] Peter Scheuermann, Gerhard Weikum, and Peter Zabback. Adaptive load balancing in disk arrays. In *Foundations of Data Organization and Algorithms*, pp. 345–360, 1993.

[4] Mong Li Lee, Masaru Kitsuregawa, Beng-Chin Ooi, Kian-Lee Tan, and Anirban Mondal. Towards self-tuning data plaement in parallel database systems. *SIGMOD Record*, Vol. 29, No. 2, pp. 225–236, Sep. 2000.

[5] Hisham Feelifl, Masaru Kitsuregawa, and Beng-Chin Ooi. A fast convergence technique for online heat-balancing of btree indexed database over shared-nothing parallel systems. In *11th Int'l Conf. on Database and Expert Systems Applications*, Sep 2000.

[6] Hui-I Hsiao and David DeWitt. Chained Declustering: A new availability strategy for multiprocessor database machines. In *Proceedings of 6th International Data Engineering Conference*, pp. 456–465, 1990.

[7] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pp. 441–448, Nov. 1999.

[8] Teradata Corp. *DBC/1012 Database Computer System Manual Release 2.0*, document no. c100001-02 edition, November 1985.

[9] 渡邊明嗣, 横田治夫. 値域分割された分散ストレージにおける効率的なアクセス負荷の記録と管理. Technical Report DE2003-116, DC2003-29, 電子情報通信学会, 2003.

[10] David J. DeWitt, Robert H. Gerber, Goetz Graefe, Michael L. Heytens, Krishna B. Kumar, and M. Muralikrishna. Gamma - a high performance dataflow database machine. In Wesley W. Chu, Georges Gardarin, Setsuo Ohsuga, and Yahiko Kambayashi, editors, *VLDB'86 Twelfth International Conference on Very Large Data Bases, August 25-28, 1986, Kyoto, Japan, Proceedings*, pp. 228–237. Morgan Kaufmann, 1986.

[11] Hisham Feelifl and Masaru Kitsuregawa. Ring: A strategy for minimizing the cost of online data placement reorganization for btree indexed database over shared-nothing machines. In *DASFAA 2001*. IEEE Computer Society, April 2001.

渡邊 明嗣　Akitsugu WATANABE
平 12 東工大・工・情工卒．平 17 同大大学院・情報理工・計算工・修士課程了．同年 同大大学院・情報理工・計算工・博士後期課程．主としてデータ工学向けの並列アーキテクチャに従事．中でも負荷分散に関する研究に主眼を置く．日本データベース学会学生会員．

横田 治夫　Haruo YOKOTA
昭 55 東工大・工・電物卒．昭 57 同大大学院・情報・修士課程了．同年富士通 (株)．同年 6 月 (財) 新世代コンピュータ技術開発機構研究所．昭 61(株) 富士通研究所．平 4 北陸先端大・情報・助教授．平 10 東工大・情報理工・助教授．平 13 東工大・学術国際情報センター・教授．工博．主としてデータベース，データ工学向けの並列アーキテクチャ等に関する研究に従事．日本データベース学会，電子情報通信学会，情報処理学会，人工知能学会，IEEE，ACM 各会員．