Scanning Text and BLOBs Efficiently
Or Testing Performance Meaningfully

By David Adams

TN 05-42

Overview

文字列、テキスト、BLOB の内容をバイト単位でスキャンすることを要求するプログラムタスクには様々なものがあります。たとえば、大文字と小文字を区別するような文字列の比較、カスタム解析、ピクチャ、BLOB、ドキュメントの比較などにはバイト単位のスキャンが関係します。テキストや BLOB の最適な比較法については、デベロッパの目的、意図、経験、好みによって意見が分かれるところです。しかしながら、実際にテストを繰り返した結果を提出しないで交わされる意見は単なる見解に過ぎません。このテクニカルノートでは、テキストおよび BLOB をスキャンするための複数の方法について、テスト結果を出力するサンプルデータベースを提供しています。このテストシステムは、最適の方法を見つける上で役に立つだけではなく、気をつけていないとテスト結果に惑わされて、誤った結論に至りかねない点をも例証しています。このテクニカルノートを読むことによって、以下の事柄についての理解を深められるはずです。

- テキスト/BLOB の比較をする際に関係する時間とメモリ使用量のバランス。
- 複数の実用的なスキャニングコードの開発方法および運用方法。
- パフォーマンスに大きな差を生じさせるポインタ特有の問題。
- テスト結果を解析または利用する際に考慮すべき点。

このテクニカルノートに含まれているサンプルテストデータベースに加え、テクニカルノート 05-41 Case-Sensitive Operations in 4th Dimension の内容も参考にすることができます。

Scanning: Small Differences Accumulate

コードのパフォーマンス特性をよく調べるべき理由として、次のテスト結果について考えてみましょう。これは 32000 バイトのテキストをスキャンした結果をまとめたものです。

データの種類	比較の方法	所要時間
テキスト	元のテキストを直接スキャン	2
テキストへのポインタ	元のテキストを複製して直接スキャン	2
テキストへのポインタ	元のテキストをポインタでスキャン	710
BLOB	元の BLOB を直接スキャン	1
BLOB へのポインタ	元の BLOB を複製して直接スキャン	1
BLOB へのポインタ	元の BLOB をポインタでスキャン	4

驚くべきことに、ポインタを介してテキストをスキャンすると、同じデータの BLOB を直接スキャンするよりも 700 倍の時間がかかることが分かります。(このテストが問題の全体像を伝えていない点については後で論じます。)このような発見は予想外のものであり、簡単には説明がつかないものです。このテスト結果や他のテスト結果について考察する前に、方法による速度の違いをもたらす設定項目や基本的な要素についてまとめておきましょう。

About the Test Database

上記のテストに使用したサンプルデータベースはこのテクニカルノートと共に提供されています。運用環境と同じ環境でテストを実行すれば、それだけ意味のあるデータを得ることができます。その環境でパフォーマンスを向上するような修正を加えることもできるかもしれません。

About the Test Results

このテクニカルノートで紹介しているテスト結果は、すべて 4th Dimension 2004.2 で実際に実行したテストの結果に基づいています。特に明記していない限り、所要時間はもっとも良い結果を 1 とした相対値で示されています。たとえば、上記のテストではポインタを介して BLOB をスキャンするのに要した時間は BLOB を直接スキャンした場合の 4 倍であることを意味しています。(より広範なテストの結果では 1:1.67 のような比率になります。)ミリ秒で表わしたテスト結果を知りたいのであれば、実際の運用環境でテストを実行してください。

注記 値はすべてコンパイルモードにおける実行結果に基づいています。テキストや BLOB の処理はコンパイルによって大幅に速度が向上するため、インタプリタモードにおける動作は考慮に入れていません。なお、範囲チェックの有無による速度差はほとんどありません。

Review: Syntax for Scanning Text and BLOBs

文字列またはテキストの中にある 1 文字を読み取ることは、概念的に BLOB の 1 バイトを読み取ることと同一です。しかし、4th Dimension における表記と番号の振り方が文字列/テキストと BLOB では異なっています。次の表では、それぞれ 32000 バイトのテキストと BLOB が比較されています。

	表記	最初	最後	サイズ
テキスト	text[[1]]	1	32000	32000
BLOB	BLOB{0}	0	31999	32000

- 文字列/テキストは 1 文字目からの位置で指定しますが、BLOB はバイト 0 からのオフセットで指定します。したがってテキストの第 1 バイトは位置 1 番であり、BLOB の第 1 バイトは位置 0 番です。
- Length 関数で返される文字列/テキストのサイズは文字数と一致します。BLOB size で返される BLOB のサイズはバイト数と一致します。しかしながら BLOB の中にある最後のバイトの位置は BLOB size-1 となります。
- 文字列/テキストの中から 1 バイトを読み取るためには[[]]を使用します。
- BLOB の中から 1 バイトを読み取るためには{}を使用します。つまり配列の要素番号を指定する際に使用するのと同じ記号を用います。BLOB は、バイトの配列であると考えることができるかもしれません。

Review: Passing Parameters by Value or Pointer

4th Dimension では、パラメータを値またはポインタで渡すことができます。次の表では、テキストまたは BLOB を解析するルーチンに渡す際のパラメータ表記がまとめられています。

データの種類	渡し方	スキャンの対象	表記例
テキスト	値	パラメータで受け取ったコピー	$1[[$character_index]]
テキストポインタ	ポインタ	ポインタを介して元のデータ	$1->[[$character_index]]
BLOB	値	パラメータで受け取ったコピー	$1{$byte_index}
BLOB ポインタ	ポインタ	ポインタを介して元のデータ	$1->{$byte_index}

値あるいはポインタでパラメータの受け渡しをすることには、それぞれメリットとデメリットがあります。値で渡す場合、4th Dimension は元のデータをパラメータにコピーします。したがって余計にメモリを使用します。たとえば次のコードでは BLOB を直接渡しています。

ScanTest_BlobParameter (ScanTest_Sample_blob)

ScanTest_BlobParameter メソッドでは、ScanTest_Sample_blob 変数が$1 にコピーされます。ScanTest_Sample_blob がクリアされたとしても、*ScanTest_BlobParameter* メソッドの中で$1 の内容は変わりません。パラメータに値を直接渡すことのメリットは、そのデータをメソッドの中で直接使用することができるという点です。

$1{$byte_index}

逆にデメリットは余計なメモリを使用するという点です。BLOB が 2MB であれば、少なくとも 4MB のメモリが使用されることになります。メモリサイズが問題なのであれば、元のデータに対するポインタを渡すという方法もあります。たとえば次のコードでは BLOB をポインタで渡しています。

ScanTest_BlobPointer (->ScanTest_Sample_blob)

ScanTest_BlobPointer メソッドでは、ScanTest_Sample_blob 変数に対するポインタが$1 に代入されます。ScanTest_Sample_blob がクリアされれば、$1 はクリアされたデータを指すことになります。ポインタを渡すことのメリットは、メモリの使用量が少ないという点です。デメリットは、ポインタの参照先をたどるために処理時間がかかるという点です。一回の処理い要する時間は微々たるものですが、これが何千、何万回も発生すると大きなものになります。

$1->{$byte_index}

注記 4th Dimension のポインタは、C 言語などのポインタとは内部構造が異なっており、参照先を調べる際の負担がかなり重くなっています。

Results Revisited

メモリやパフォーマンスに関する前述の点を踏まえた上で、幾つかのテスト結果について考えてみましょう。値は 32000 バイトのテキストおよび BLOB を 50 回スキャンした結果の平均値です。すべてコンパイルモードで実行され、最低値と最高値の幅は比較的少ないものでした。

データの種類	比較	速度
BLOB へのポインタ	元の BLOB を複製してスキャン	1
テキストへのポインタ	元のテキストを複製してスキャン	2
BLOB へのポインタ	元の BLOB をポインタでスキャン	4
テキストへのポインタ	元のテキストをポインタでスキャン	710

結果を概観し、次のような結論に至るかもしれません。

- ポインタによる操作は効率が悪い。直接値を渡す場合と比較して、テキストでは 355 倍、BLOB では 4 倍も時間がかかる。したがってポインタは使用するべきではない。
- テキストによる操作は効率が悪い。同じサイズの BLOB と比較して、直接値を渡す場合は 2 倍、ポインタを使用する場合は 180 倍も時間がかかる。したがってテキストは使用するべきではない。

以上の結論は、実際のテスト結果とも合致しており、理にかなっているように思えます。しかし、ここには落とし穴があります。テスト結果は正確なものですが、完全なものではないからです。次に、他のテスト結果を調べ、それをどのように利用するべきかを考慮してみましょう。

Tests Results Are Easy to Overapply

ベンチマークやテスト結果は、議論を展開する上で非常に説得力にあるデータです。残念なことに、不完全または間違ったテスト、あるいは間違った理解の結果として、次のような落とし穴に陥ることがあり得ます。

- 不正確または無意味なデータに頼る。
- テスト結果によって先入観が強まる。
- 特殊なケースで実行したテスト結果があらゆる環境に適用できると考える。
- テスト結果の足りない点を見落とす。

こうした危険があるとはいえ、テストの実施は非常に価値があるものです。理想をいえば、テストおよびその結果は複数の人間によって解析されるべきです。ひとりのプログラマがテストの方法や解釈に影響を及ぼしている先入観や問題をすべて見抜くことには限界があります。

このテクニカルノートで紹介したテストの方法や結果について、気づいた点や洞察があれば、dpadams@island-data.com までご連絡ください。

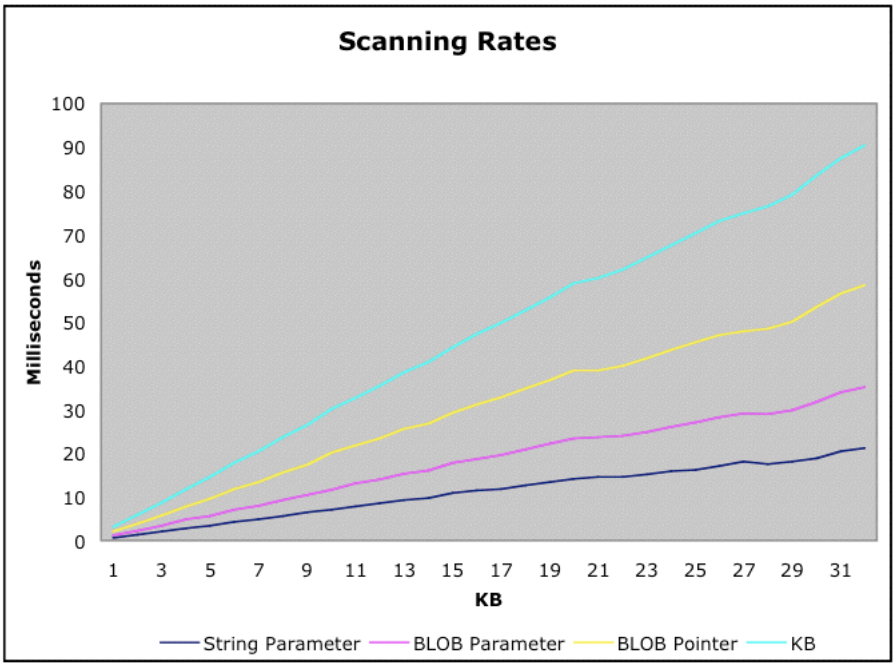
さて、テストに潜む様々な危険について論じたところで、別の観点に立って実施したテストについて考えてみましょう。

How Slow Is It to Read Text Through a Pointer?

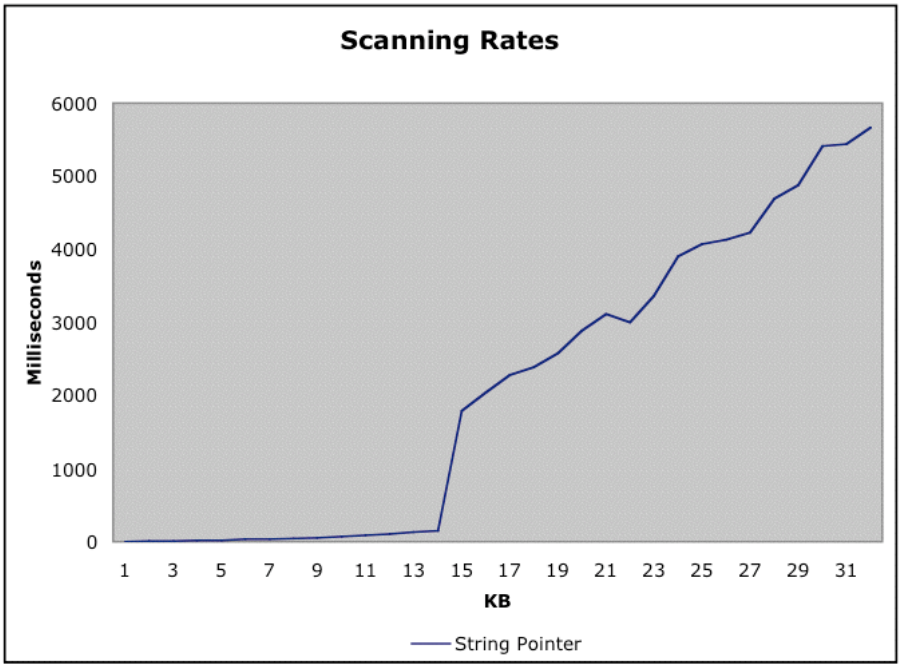
次のテスト結果は、これまでと同じテストを 32000 バイトの代わりに 1000 バイトで実施したものです。ここでは最速の結果を 1 として相対値で所要時間が示されていますが、結果が近接していたために、比率は小数点以下まで表記されています。

データの種類	比較	所要時間
BLOB へのポインタ	元の BLOB を複製して直接スキャン	1.00
BLOB へのポインタ	元の BLOB をポインタでスキャン	1.45
テキストへのポインタ	元のテキストを複製して直接スキャン	5.69
テキストへのポインタ	元のテキストをポインタでスキャン	7.25

このテストでは、方法によるスピードの相対関係を調べているので、前回のテストと似た結果が返されそうなものですが、実際には違う結果になっています。もっとも顕著な違いは、1000 文字のテストの場合、テキストポインタは 7 倍遅いのに対して 32000 文字のテストではその値が 700 倍になっている点です。通常、値に 2 桁の差が出るような場合、テストの方法自体を疑う必要があります。テストの方法に問題がないようであれば、何が起きているのかを詳しく調べるために他のテストを考える必要があるかもしれません。今回のテストの場合、32000 文字のスキャンには方法に関係なく 1000 文字の 32 倍の時間がかかるものと推測できます。この仮定は、元のテストコードを使用して簡単に検証することができます。下の表は、1000 文字、2000 文字、というようにデータサイズを増やした場合の結果です。完全なシーケンシャルスキャンに要する時間はバイト数に比例することが分かります。



グラフには、テキストの直接渡し、BLOB の直接渡し、BLOB のポインタ渡しについての結果が表わされています。テキストのポインタ渡しについては、次のグラフのとおりです。



途中まではサイズに比例して所要時間が長くなっていますが、14000-15000 バイトのあたりで急激な増加が起きています。テキストをポインタ経由でスキャンする際に要する時間は、テキストのサイズによって大きく変動するということが分かります。

注記 使用するマシンや 4th Dimension のバージョンによって、急激な変化が起きる位置は変動し、場合によってはまったく起きないこともあります。

The Danger of Incomplete Testing

この例から分かるように不完全なテストには危険が潜んでいます。使用するデータのサイズによっては、同じテストであっても、異なる印象を与えるものになる可能性があります。10000 バイトでテストしたデベロッパは、ポインタの使用には問題がないと考えても、20000 バイトでテストしたデベロッパは、ポインタは使えないと判断するかもしれません。このように不完全なテストでは不十分です。テストを実行する際には、幅広いデータ値を使用することを心掛けてください。数値やグラフには説得力がありますが、テストに含まれないデータがあることまでは表に現れないからです。

Why Does the Text Scanning Rate Vary?

それにしても、なぜテキストポインタでは突然、値が変動したのでしょうか。ほんとうの答えを知るには 4th Dimension の内部構造を調べるしかありません。外的なテストで調べられることには限界があるからです。原因については、様々な憶測をすることができるかもしれません

が、結局のところ、それらは問題ではなく、要は再現可能な動作に対してどのように対処するかという点が重要です。

Numbers Are Meaningless without Context

テストの結果をみると、結論は明白だと考えるかもしれませんが、性急に判断するよりも、問題の全体像をよく把握することによって、最良の方法を決めることができます。たとえば、当初に下した結論について考えてみましょう。

- ポインタは使用するべきではない。
- テキストではなく BLOB を使用するべきだ。

データの裏付けがあるとしても、上記のような結論には問題があるといえる理由は幾つも挙げることができます。

- ポインタはデータの複製をせず、メモリの節約になる。大きなサイズの BLOB であれば、これは重要な点である。ある程度の時間がかかっても、それだけの価値があるかもしれない。すべては運用環境とシステム全体に対する顧客の要件にかかっている。
- ポインタは 4th Dimension で汎用性のあるコードを作成するための優れたテクニックである。メンテナンスの費用と開発の労力を節約できることには大きなメリットがある。
- BLOB が速いとしても、テキストほど簡単には扱えない事情がある。Position 関数など、BLOB では使用できないネイティブコマンドもある。開発が快適になり、コードが読みやすくなるのであれば、幾らかのスピードは犠牲にする価値があるかもしれない。

方法の違いによる特徴は、かなりはっきりとしていますが、どの方法にもメリットとデメリットがあり、スピードとメモリ、その他の要素についてどのようにバランスをとるかについては明確な答えがありません。望ましいのは、運用環境とユーザの要求に合った方法、すなわち特定のパフォーマンスベンチマークやシステム要件に見合ったアプローチを採択することです。

How Fast Is Fast?

典型的な環境では、ポインタ使用よりも値の直接渡しのほうが若干、速いわけですが、ここで「速い」と「遅い」の定義について少し考えてみましょう。表のデータは、方法同士の相対的な速度差であり、実際の計測時間ではありません。ある方法が別の方法よりも 1000 倍速いといっても、他方の結果が 1000 分の 1 秒であり、繰り返しの処理をしないというのであれば、実際、その比較には意味がありません。それに、処理時間以上には時間を節約できません。1 秒の処理は、いくら最適化しても 1 秒以上の節約にはならないのです。BLOB をスキャンする場合、最初のテーブルに登録された BLOB フィールドをポインタでスキャンする場合、直接値を

渡すよりも 4 倍の時間がかかります。(前述したように、様々なテストの平均的な結果は 1.67 倍です。)4 倍は大きな差のように感じられますが、今日の平凡なマシンで 1MB の BLOB を使用した場合、その違いは 2 秒に満たないものです。わずか 1 秒の処理時間を短縮するために 1MB のメモリを余計に使う必要があるでしょうか。それは処理の種類、ユーザの有無によって決まる問題です。無人の処理ではあまり問題がなくても、画面の再描画を待つユーザにとって 2 秒はけっこうな時間です。テストの結果は、運用環境と関連させて分析してください。

Available Memory and Operations on BLOBs

テストでは、一貫して BLOB を直接渡すよりもポインタを介したほうが時間がかかるという結果になりました。したがって、スピードを求めるのであれば、ポインタを使用するよりも直接値を渡したほうが良いように思えます。しかし、BLOB の場合は注意が必要です。値を渡すということは、データがコピーされることを意味し、4th Dimension にメモリの配分を要求することになります。サイズが大きければ、かなりメモリ量が関係してきます。4th Dimension 2004 が動作するからには、ある程度のメモリ容量を持つマシンが使用されているはずですが、それでもメモリは無尽蔵にあるわけではありません。実際、メモリを確保するために他のデータを移動する必要が生じれば、かなりのパフォーマンスダウンになります。したがって、ポインタを使用したほうが速く処理が完了する場合もあります。テストによってこの点を確証することもできますが、それよりも差し迫った問題として、4th Dimension はメモリが足りないと BLOB 操作の途中で強制終了するかもしれません。

Personal Recommendation

個人的な見解として、特別な条件がない限り、スキャン処理の目安として私は次の基本法則を推奨します。

- テキストは、パフォーマンスの壁にあたる可能性があるので値を直接渡すこと。
- BLOB は、余分な RAM の使用を避けるためにポインタで渡すこと。

これらはデフォルトの基本方針であり、絶対的なものではありません。他の法則を用いても構いませんし、特定の状況では、明らかに時間とメモリを節約する他の方法が選択できます。

Summary

このテクニカルノートでは、テキストや BLOB をスキャンする際に値を直接渡す方法とポインタを使用する方法のスピードとメモリ使用量の差について論じました。テストサンプルデータベースは、テキストや BLOB をスキャンするためのコードをデザインする際に最適な方法を見極める参考にすることができます。テストシステムおよびテスト結果を例にして、パフォーマンステストに潜む問題についても考察しました。