READ ONLYデータベースによる、
高速・大規模・低コストの
検索・ダウンロードサービス実現の試み

２０２０年２月１４日

「巨大時系列データの高速アクセスに関する研究」チーム
古庄 晋二・生座本 義勝・山本 幸生・早部 秀一

背景：経緯

- Apollo 11～17号が月に設置した地震計データのダウンロードサービスの高速化に取り組んだ。
- ビッグデータからサブビッグデータを抽出する処理である。
- このための良いソリューションが見当たらず、今回、その解決を図った。

背景：「Read Only」という救い

- Apolloのデータベースは、
  - A)　「データの蓄積・管理」サブシステムと、
  - B)　「**オンライン検索・分析・ダウンロード**」サブシステムに、

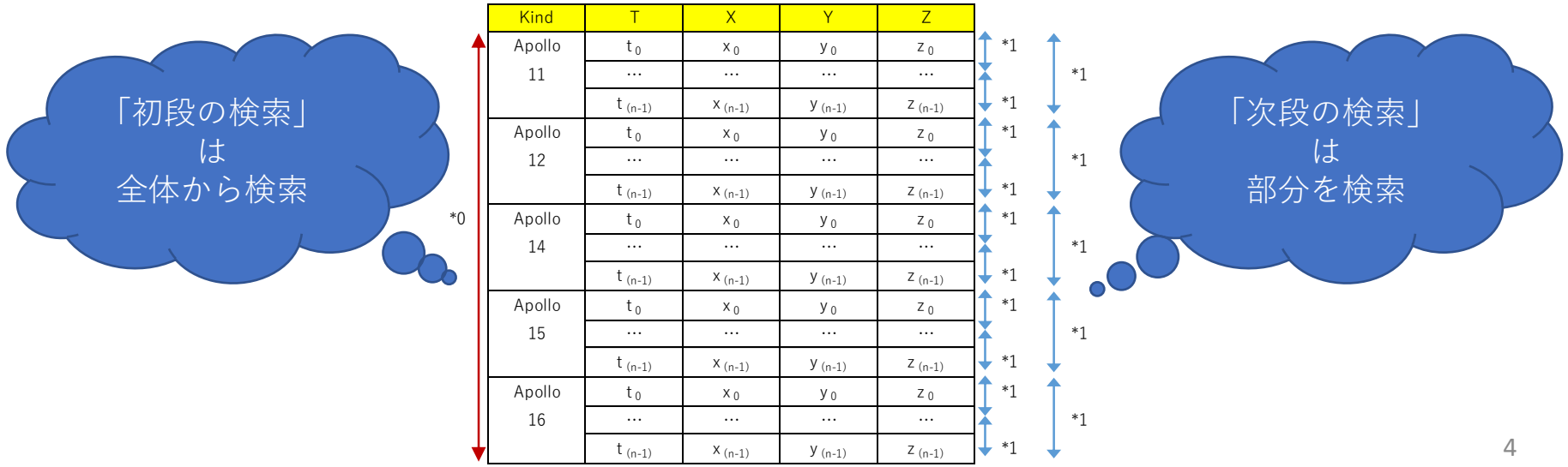
分割できる。

- A に使用されるデータベースは処理時間の制約が少なく、実現が可能。
- 一方、B のデータベースは高速なレスポンスが要求されるが、Read Only で構わない。

⇒　Read Only 前提だと、高速・大規模を実現する解法がある。

検索の高速化：２段階の検索が必要

- 「初段の検索」検索対象を絞り込む
  - 例：Apollo 12号の１月のデータの中から・・・
- 「次段の検索」初段で絞り込まれた範囲を、多項目の範囲条件で検索する
  - 例： 10≦ x ≦12, 20≦ y ≦22, 30≦ z ≦32
- 「初段」、「次段」ともに既存のインデックスでは低速。



Kind	T	X	Y	Z
Apollo 11	t_0	x_0	y_0	z_0
	…	…	…	…
	$t_{(n-1)}$	$x_{(n-1)}$	$y_{(n-1)}$	$z_{(n-1)}$
Apollo 12	t_0	x_0	y_0	z_0
	…	…	…	…
	$t_{(n-1)}$	$x_{(n-1)}$	$y_{(n-1)}$	$z_{(n-1)}$
Apollo 14	t_0	x_0	y_0	z_0
	…	…	…	…
	$t_{(n-1)}$	$x_{(n-1)}$	$y_{(n-1)}$	$z_{(n-1)}$
Apollo 15	t_0	x_0	y_0	z_0
	…	…	…	…
	$t_{(n-1)}$	$x_{(n-1)}$	$y_{(n-1)}$	$z_{(n-1)}$
Apollo 16	t_0	x_0	y_0	z_0
	…	…	…	…
	$t_{(n-1)}$	$x_{(n-1)}$	$y_{(n-1)}$	$z_{(n-1)}$

検索の高速化：初段の検索の課題

- 大規模データの必要領域を素早く検索したいが、、
  - 範囲の検索に対応する良いインデックスがない
    - 範囲なのでハッシュは使えない
    - B-treeの発案者であるRudolf Bayerは次のように発言している

      If the world was perfectly static, such a performance could also be achieved with other indexing techniques.

  - 膨大な検索結果集合を一瞬で記録できない
    - Read Only なら解決策がある。
      検索結果集合の空間のヒット区間を記憶するだけで良い。
    - ベンチマークでは、４０億行のテーブルから５億行を検索するのに１ミリ秒以下で済んだ。

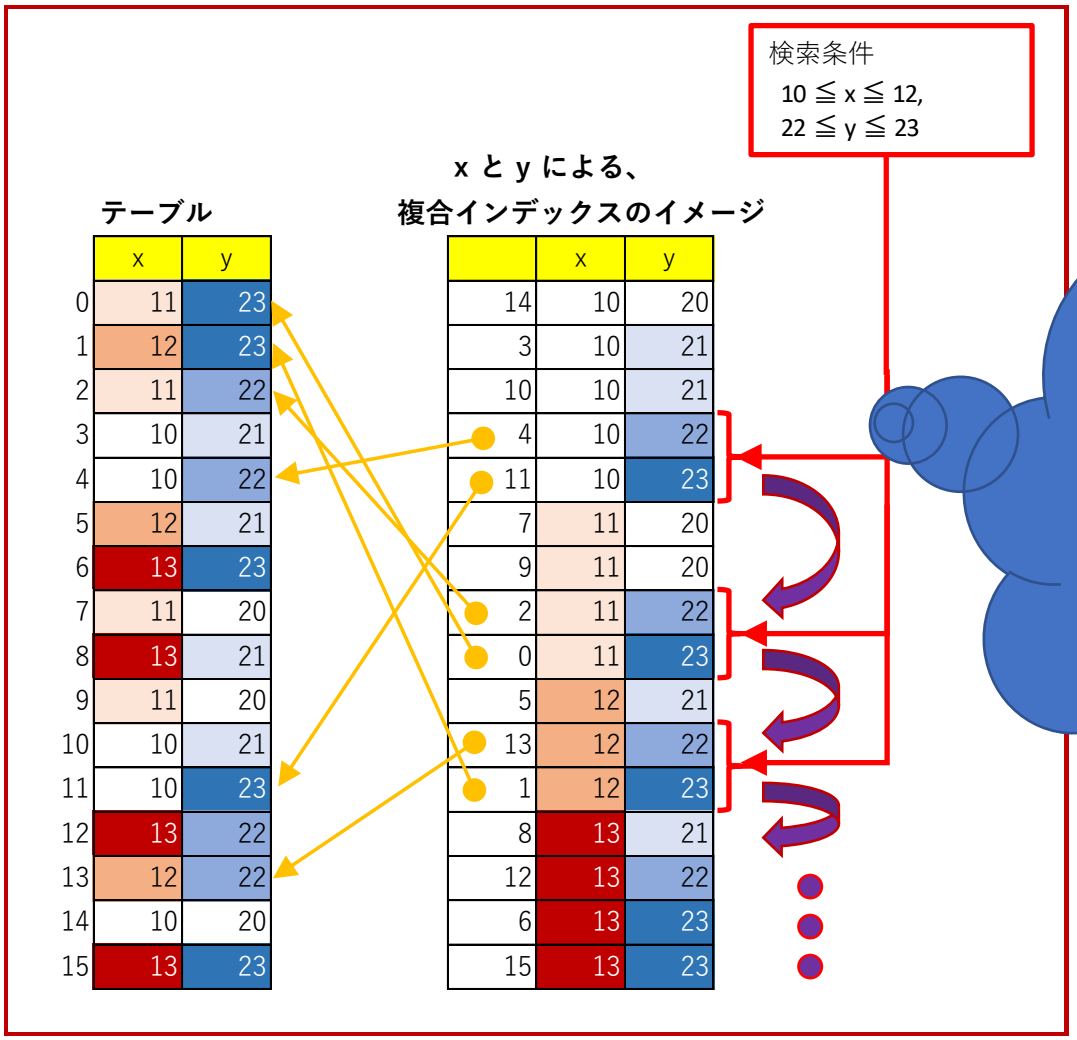
検索の高速化：次段の検索の課題

- 既存インデックス技術では、
  - 良い部分インデックスがない
    - 多数の範囲をカバーしにくい
    - コンパクトにしにくい
  - 複数項目の範囲の検索が遅い
    （複合インデックスでは複数項目の範囲の検索は遅い　⇒　次ページ）

複数項目範囲インデックス

- Read Only なら解決策がある。
  - （Read Only前提の）転置インデックスをカスケードする手法で
  - ベンチマークでは、1000万行のテーブルから、x/y/z座標の範囲を検索すると、16ミリ秒～

※　複合インデックスを用いて複数項目の範囲を検索すると、
インデックス上のヒット区間が分散する。
その結果、インデックスの重大なパフォーマンス低下が発生する。



複合インデックス上のヒット区間の分散は、
インデックスが１次元の大小関係に基づいて設計されていることに由来する。

多次元の大小関係に基づいてこの問題の解決を試みたのが、
「複数項目範囲インデックス」である。

テーブルの分割・組換：メリット

- テーブルが大規模化すると、
  - システムが大きく・高価・低速になり、
  - データの管理も難しくなる。
- テーブルを分割できると、
  - ストレージの増設だけで大規模なデータに対応できることが期待できる。
- 分割したテーブルを自由に選択し、所望の順序で仮想的にUNIONできれば、
  - ユーザ／サービス毎に最適なデータセットを作れるので、サービスの多様化・高度化ができる。

テーブルの分割・組換：
テーブル組換対応インデックス

- テーブルの分割・組換は、「仮想UNION」機能を使って行う。

「仮想UNION」機能は、１テーブル１ファイル形式のテーブルファイル群から、必要なテーブルファイルを選んで、任意の順番で仮想的にUNIONテーブルを作成する機能。

- 「仮想UNION」機能に対応するインデックスが、「テーブル組換対応インデックス」である。

  Read Only 環境が前提になる。

ベンチマーク１
初段の検索：テーブル組換対応インデックス

1. 表１に示す、Apollo 11～16号の、 項目（時間、lpx、lpy、lpz）からなるテーブル５つを「仮想UNION」し、 ４０億レコードのテーブルを作った。
2. その上で、 x/y/z 個別に検索範囲を指定し、 検索を行った。（次ページ）

表１

分類	レコード数
Apollo 11号	11,436,480
Apollo 12号	1,301,101,200
Apollo 14号	1,010,464,200
Apollo 15号	948,976,200
Apollo 16号	797,193,720
11～16号合計	4,069,171,800

スキーマ

名称	型
time	64bit整数
lpx	64bit整数
lpy	64bit整数
lpz	64bit整数

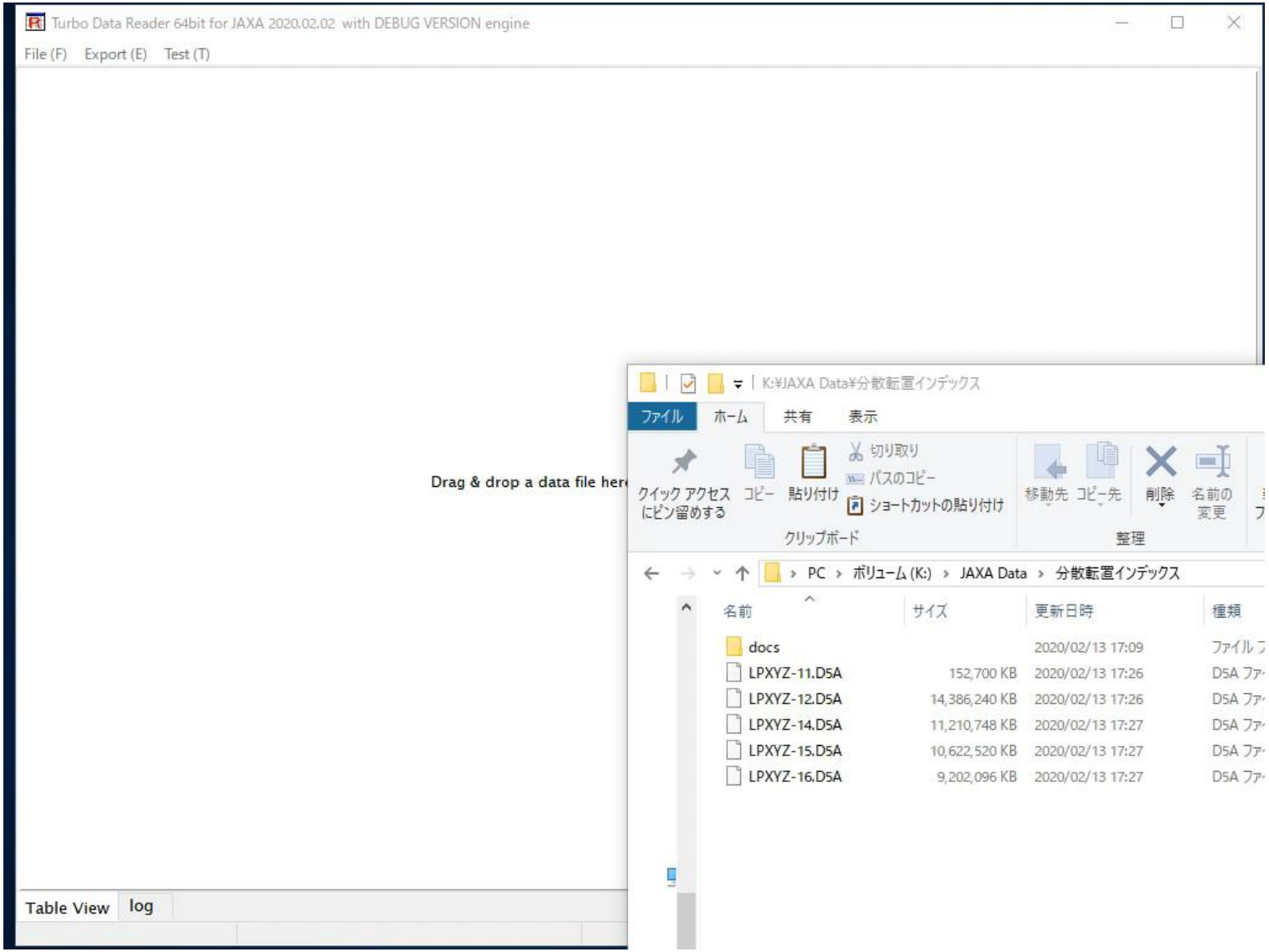
4,069,171,800行より
１項目の区間を指定して検索

	検索条件	検索前レコード数	ヒットレコード数	所要時間(mSec)	
1	517 ≦ x ≦ 519	4,069,171,800	115,229,394	1回目	0.291
				2回目	0.058
				3回目	0.055
2	522 ≦ x ≦ 522	4,069,171,800	84,781,885	1回目	0.067
				2回目	0.063
				3回目	0.103
3	520 ≦ x ≦ 521	4,069,171,800	142,464,693	1回目	0.079
				2回目	0.060
				3回目	0.080
4	514 ≦ y ≦ 517	4,069,171,800	311,907,582	1回目	0.280
				2回目	0.081
				3回目	0.078
5	524 ≦ y ≦ 526	4,069,171,800	196,663,395	1回目	0.070
				2回目	0.075
				3回目	0.081
6	523 ≦ y ≦ 527	4,069,171,800	307,228,343	1回目	0.067
				2回目	0.068
				3回目	0.080
7	502 ≦ z ≦ 504	4,069,171,800	29,213,381	1回目	0.281
				2回目	0.093
				3回目	0.089
8	514 ≦ z ≦ 518	4,069,171,800	374,244,303	1回目	0.107
				2回目	0.063
				3回目	0.072
9	518 ≦ z ≦ 523	4,069,171,800	570,110,773	1回目	0.083
				2回目	0.086
				3回目	0.067
				最大	**0.291**
				最小	**0.055**
				平均	**0.099**

４０億行から数億行を検索

最大：0.291 mSec
最小：0.055 mSec
平均：0.099 mSec

デモ１：仮想UNIONと検索
（テーブル組換対応インデックス）

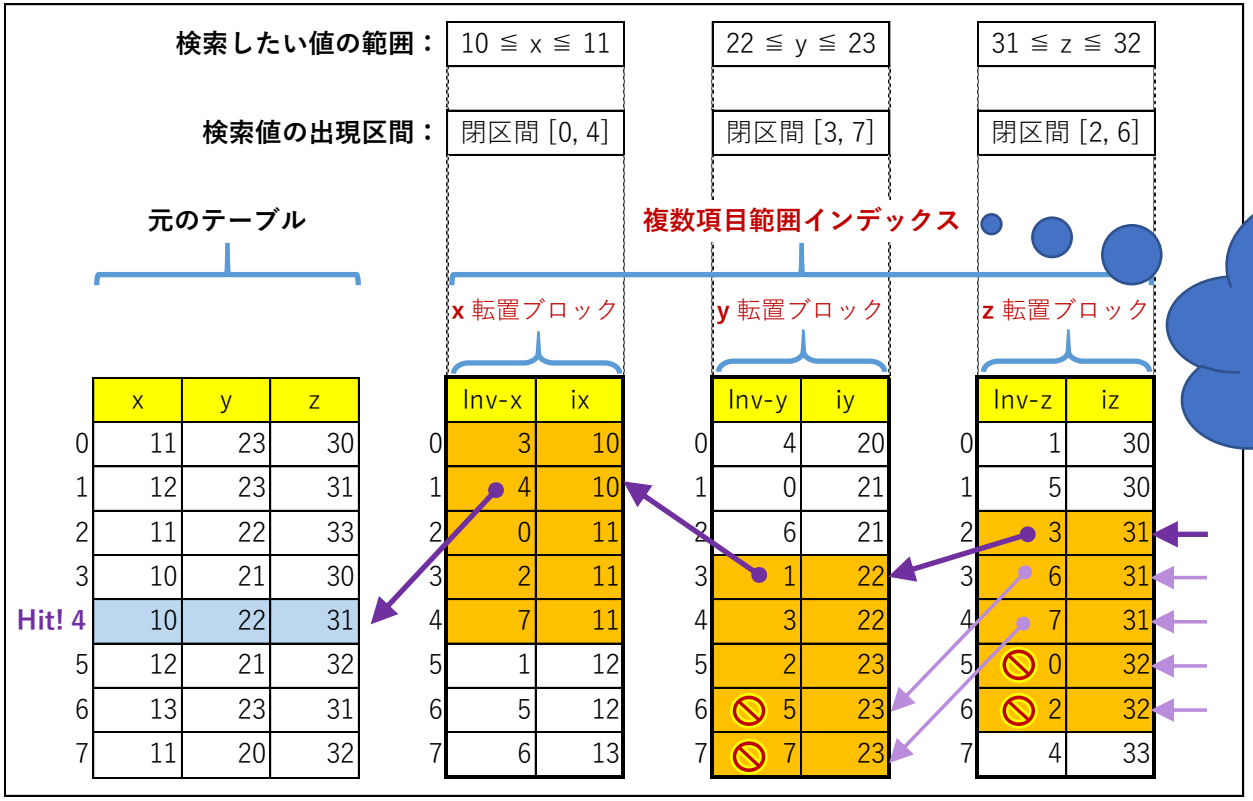


ベンチマーク２：
次段の検索：複数項目範囲インデックス

既存の「複合インデックス」では、多項目の範囲の検索は遅い。

その理由は多項目の検索を1次元の大小関係に基づくインデックスで行うため。

多項目の検索を効率よく行うには、多次元の大小関係に基づくインデックスが必要。



	x	y	z
0	11	23	30
1	12	23	31
2	11	22	33
3	10	21	30
Hit! 4	10	22	31
5	12	21	32
6	13	23	31
7	11	20	32

	Inv-x	ix
0	3	10
1	4	10
2	0	11
3	2	11
4	7	11
5	1	12
6	5	12
7	6	13

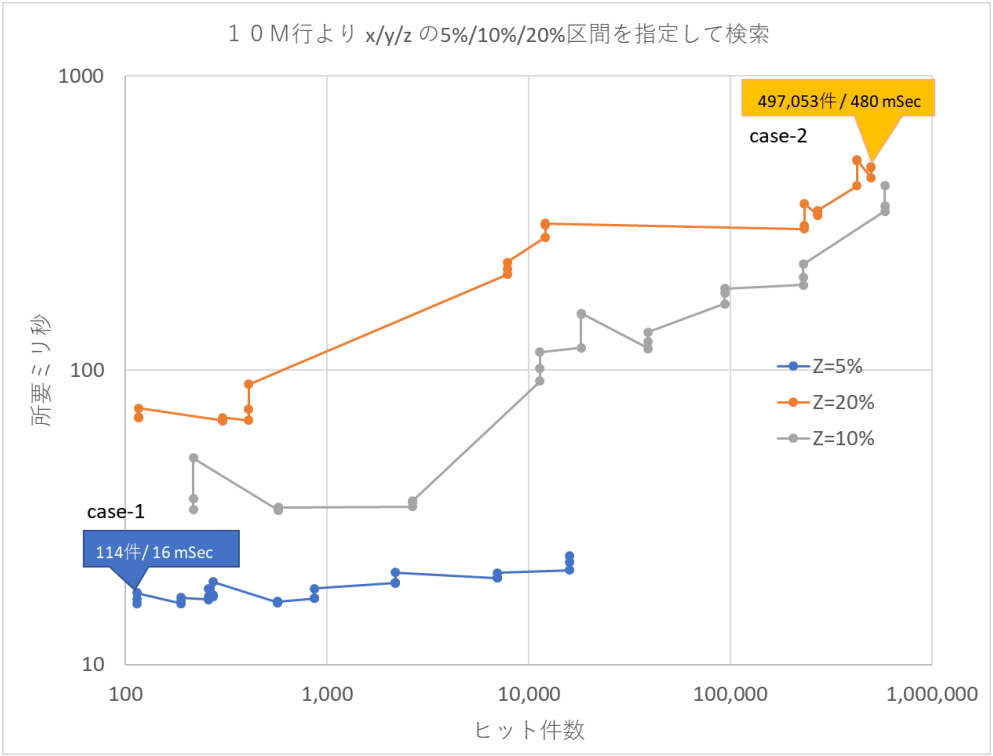
	Inv-y	iy
0	4	20
1	0	21
2	6	21
3	1	22
4	3	22
5	2	23
6	5	23
7	7	23

	Inv-z	iz
0	1	30
1	5	30
2	3	31
3	6	31
4	7	31
5	0	32
6	2	32
7	4	33

10M行より
３項目（x / y / z）の区間を指定して検索

16 mSec ～ 520 mSec

10M 00000000.D5A		X	Y	Z	Hit件数	Search mSec	Export mSec
0	1 回目	5%条件	5%条件	**5%条件**	15,940	**23.40**	44.25
	2 回目	5%条件	5%条件	**5%条件**	15,940	**20.96**	38.91
	3 回目	5%条件	5%条件	**5%条件**	15,940	**22.35**	52.29
1	1 回目	10%条件	5%条件	**5%条件**	2,184	**20.55**	10.02
	2 回目	10%条件	5%条件	**5%条件**	2,184	**19.05**	9.75
	3 回目	10%条件	5%条件	**5%条件**	2,184	**18.92**	10.11
・・・	1 回目						
	2 回目			・・・			
	3 回目						
9	1 回目	5%条件	5%条件	**10%条件**	571	**33.49**	4.47
	2 回目	5%条件	5%条件	**10%条件**	571	**33.61**	4.50
	3 回目	5%条件	5%条件	**10%条件**	571	**34.22**	4.34
・・・	1 回目						
	2 回目			・・・			
	3 回目						
18	1 回目	5%条件	5%条件	**20%条件**	303	**67.31**	3.36
	2 回目	5%条件	5%条件	**20%条件**	303	**67.78**	30.01
	3 回目	5%条件	5%条件	**20%条件**	303	**68.96**	20.96
・・・	1 回目						
	2 回目			・・・			
	3 回目						



参考文献の紹介

[1] 特許文献 US7,801,903 B2： パラレルソート

（複数項目範囲インデックスのSSDキャッシュミス低減に使用）

https://patents.google.com/patent/US7801903?oq=7%2c801%2c903

[2] 特許文献 US8,065,337 B2： パラレルソート

（複数項目範囲インデックスのSSDキャッシュミス低減に使用）

https://patents.google.com/patent/US8065337B2/en?oq=US8%2c065%2c337

[3] テーブル組換対応インデックスの動画

Zap-Over: The Big Data's Browser across Clouds,

https://www.youtube.com/watch?v=2Pg9tVocb9M&feature=youtu.be

[4] テーブル組換対応インデックス（Zap-Over）を含む解説資料

Create The Global Usage Cycle of Big Data,

https://drive.google.com/file/d/1QeJewtThOQy437x6Cib4pS4NGJlx2V1F/view

[5] 特許出願明細書（テーブル組換対応インデックス）

http://turbodata.sakura.ne.jp/Zap-Over%20Pat%20Application.pdf