PGECons
PostgreSQL Enterprise Consortium

2019年度性能トラブル対処法 インデックス設計追記版

PostgreSQLエンタープライズ・コンソーシアム　技術部会
ヤマトシステム開発株式会社

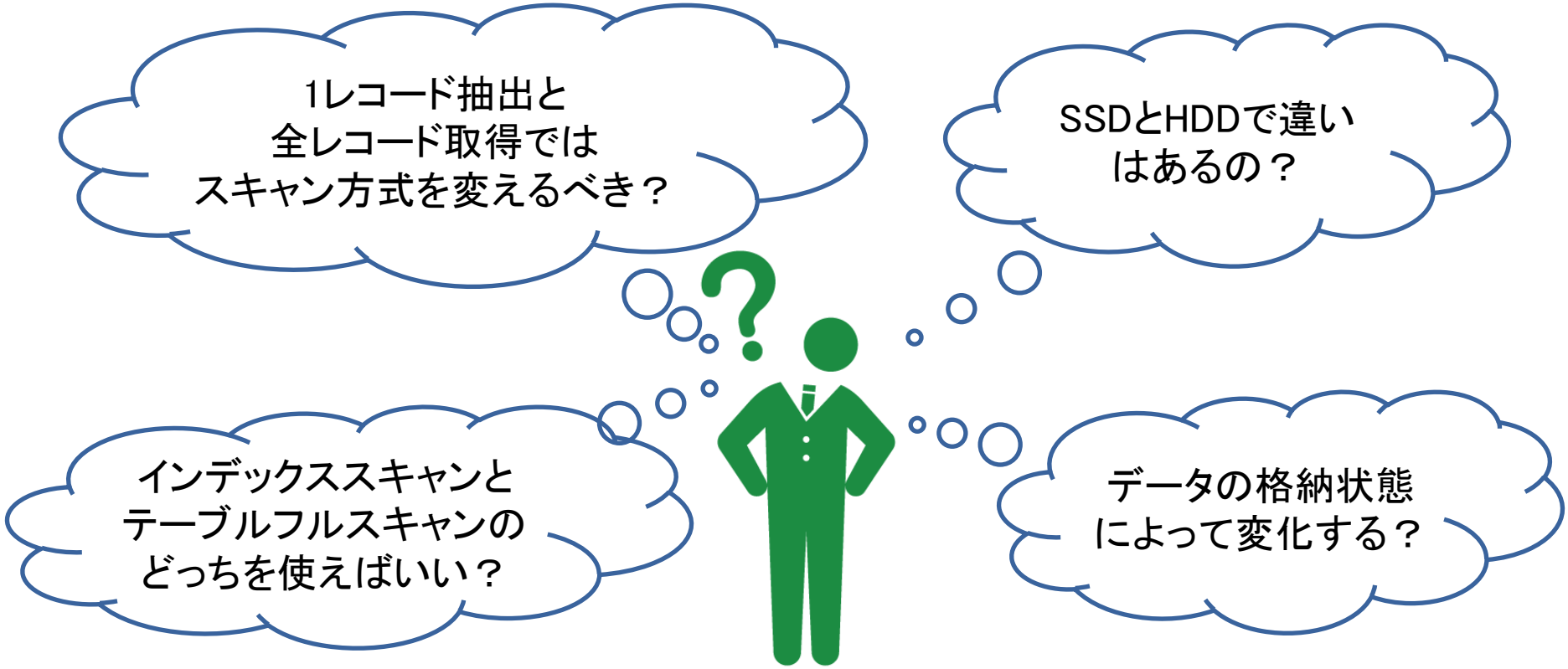
アジェンダ

- 1. 検証目的について
- 2. 検証方法と結果

1. 検証目的

1.1.疑問

- **巨大なデータベースを参照する際、どのようにテーブルからデータを取得するか悩みどころです。**



1.2.目的

- **様々な条件でデータ取得実験を行い、処理時間の傾向を分析することで、効率的にデータを取得するための要点を分析。**

2. 検証方法と結果

2.1.検証方法(１)

- 以下条件の組み合わせごとに、処理時間を計測

条件	パターン
スキャン方式	インデックススキャン
	テーブルフルスキャン
テーブルデータ配置	クラスタ化率(※)高 => 範囲検索の効率が良い
	クラスタ化率(※)低 => 範囲検索の効率が悪い
記憶媒体	HDD
	SSD（HDDの約7倍の読出し速度のものを使用）

※クラスタ化率の詳細については後述

- 抽出件数はテーブルの全件数のうち 1% ～ 100% まで 1% ごとに増加させ計測
（計測ごとに、PostgreSQLの再起動とOSのファイルキャッシュをクリア）

2.1.検証方法(２)

■ テーブルサイズ

項目	値
レコード数	1000 万件
レコード長	169 byte
ページサイズ	8,192 byte
ページ内格納レコード数	40
テーブルサイズ(※１)	1.95 GB (約25万ページ)
インデックスサイズ(※２)	0.47 GB

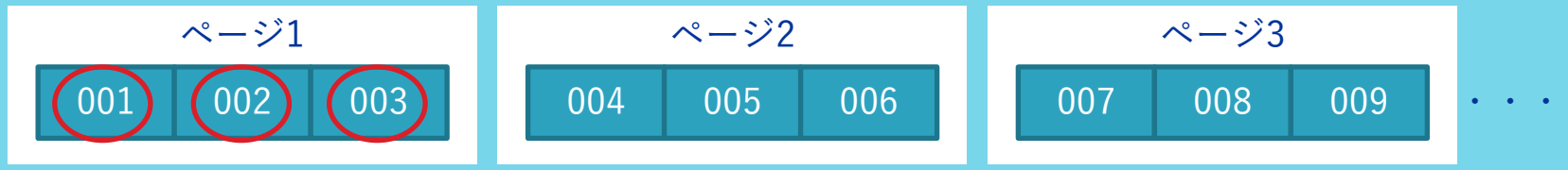
※１ shared_buffers はテーブルの全データがのる充分なサイズを設定しています。
※２ インデックスはデータ投入後に作成しています。
データ投入前に作成するとクラスタ化率低の場合インデックスサイズが大きくなり
クラスタ化率高の場合と条件が異なってしまうためです。

2.2.クラスタ化率

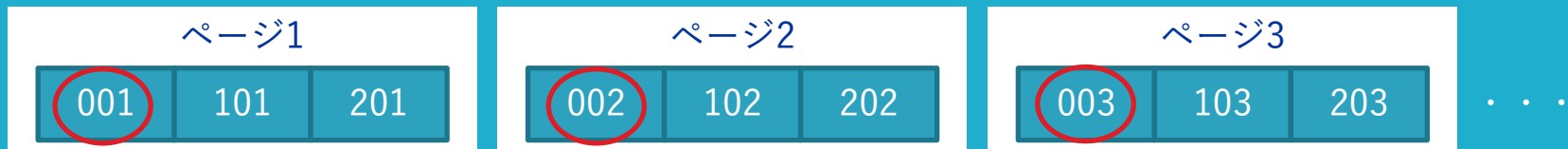
インデックスに対するテーブルのクラスタ化率	インデックス順に対するテーブルデータの物理配置	インデックススキャンによる範囲検索
クラスタ化率:高	連続して配置される	効率が良い
クラスタ化率:低	断片的に配置される	効率が悪い

インデックスデータ（データが昇順で配置）

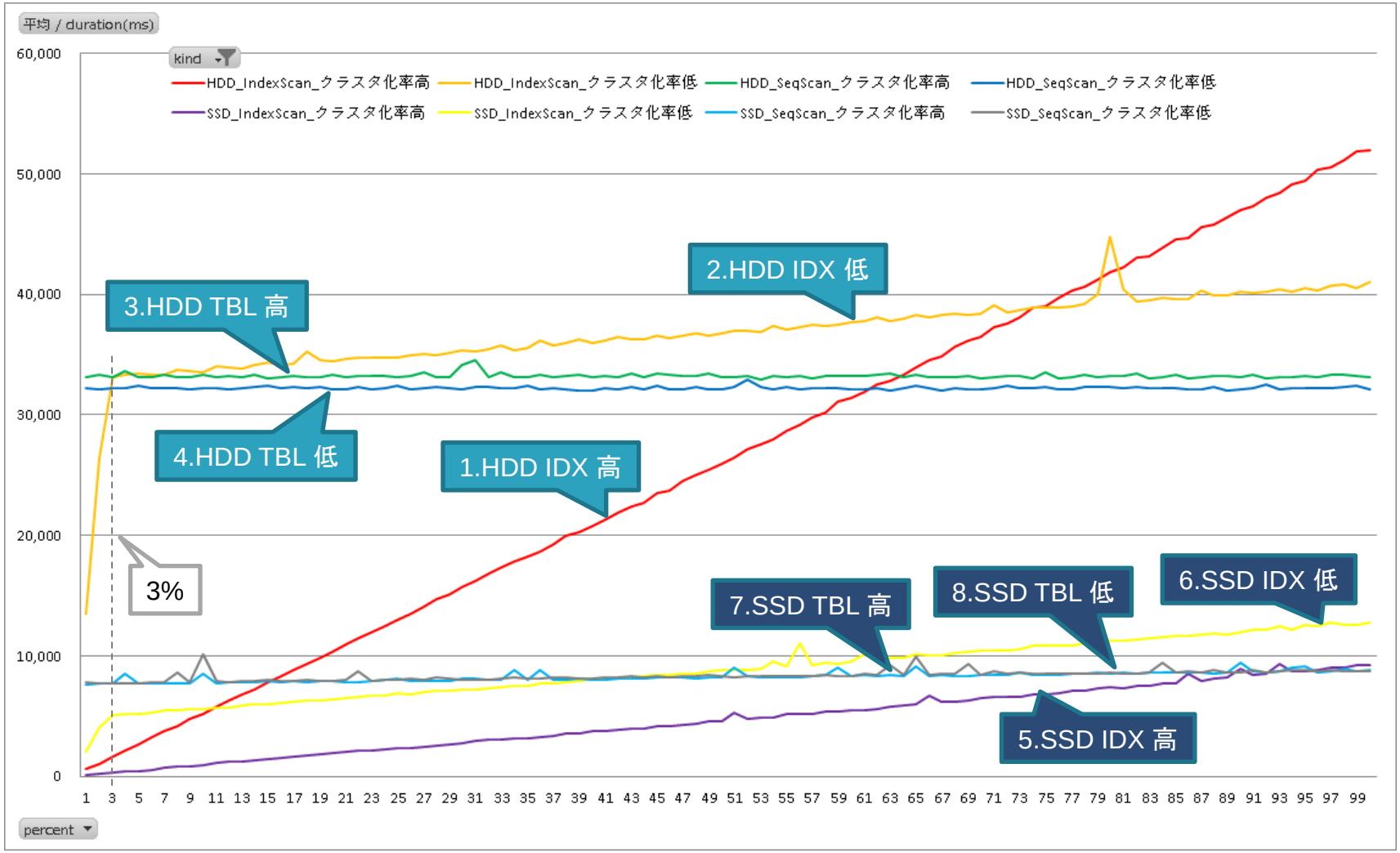
テーブルデータ（**クラスタ化率：高** → インデックス順に対し連続して配置）



テーブルデータ（**クラスタ化率：低** → インデックス順に対しデータが断片的になるよう配置）

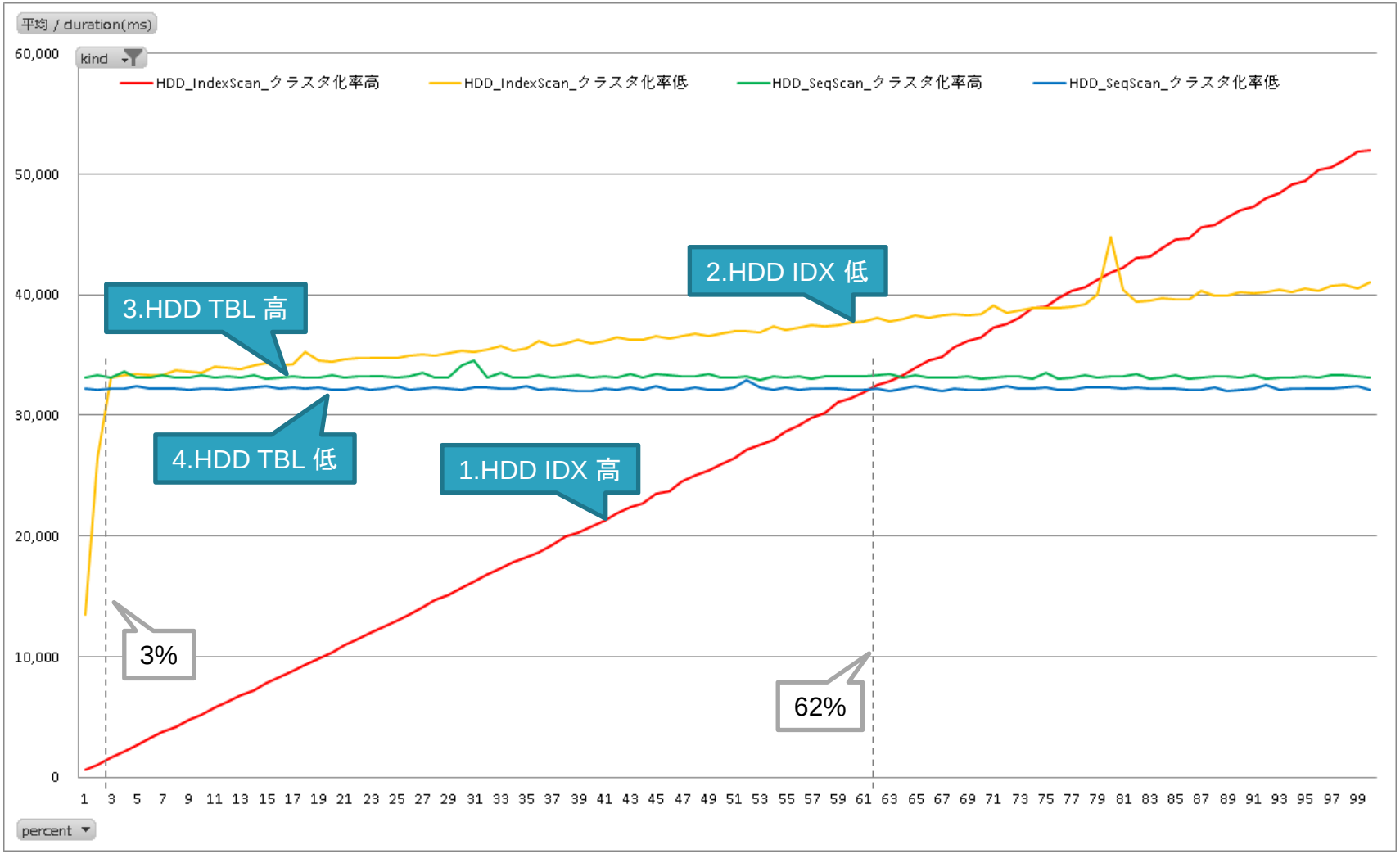


2.3.検証結果(HDD・SSD全て)



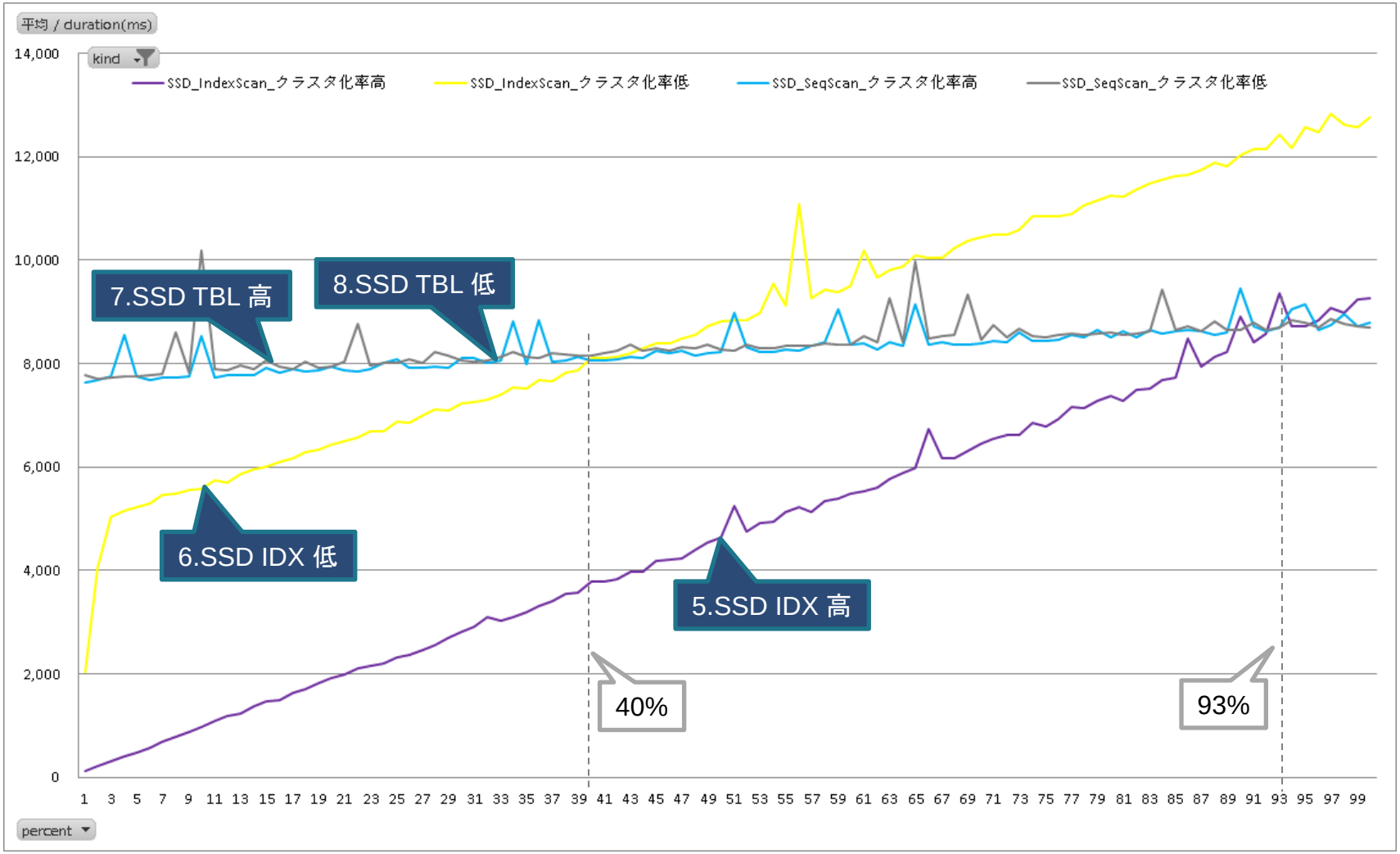
※ 横軸はデータの取得割合（1～100%）、縦軸は処理時間(ms)を示す

2.4.検証結果 (HDDのみ)



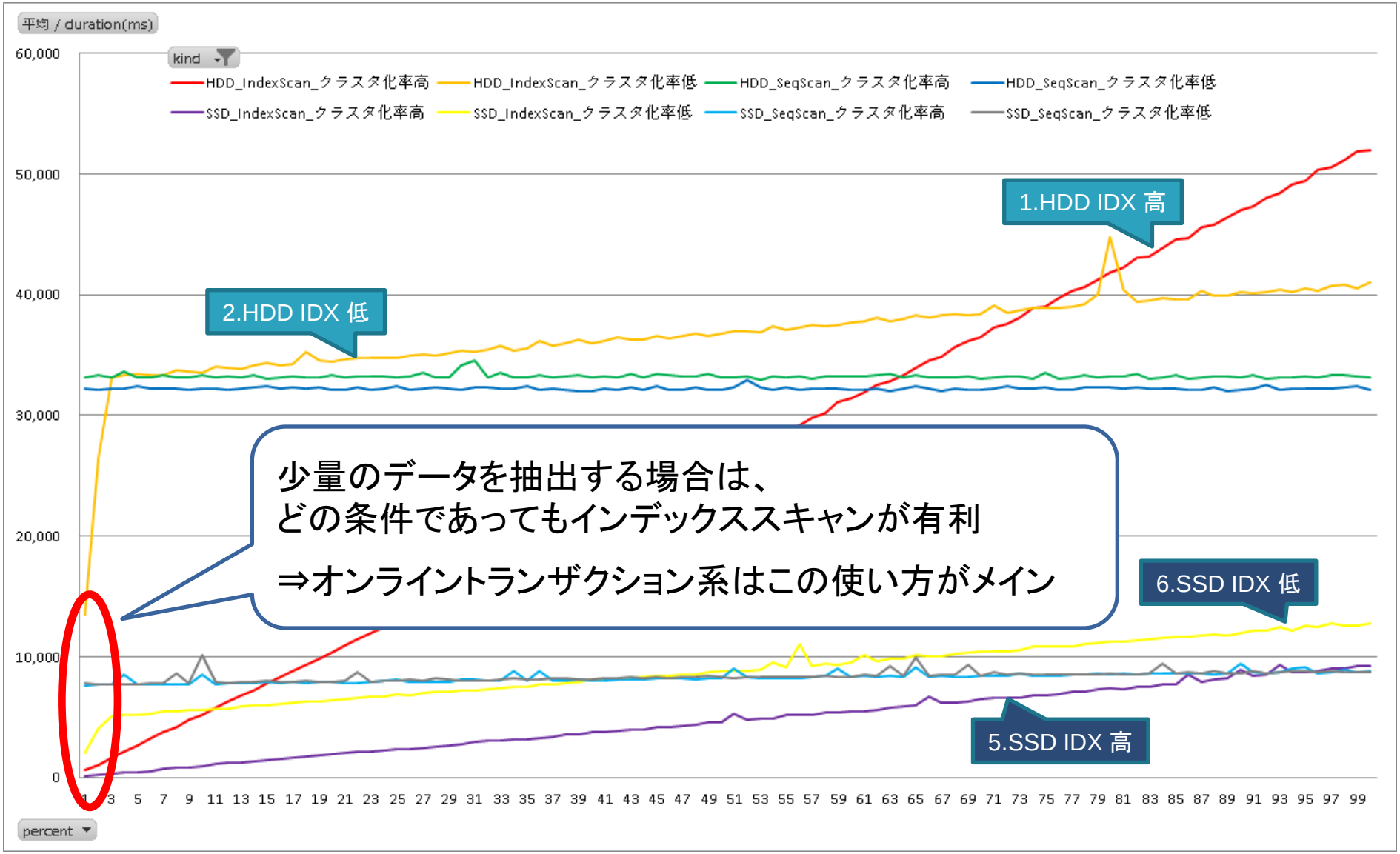
※ 横軸はデータの取得割合（1～100%）、縦軸は処理時間 (ms) を示す

2.5.検証結果 (SSDのみ)

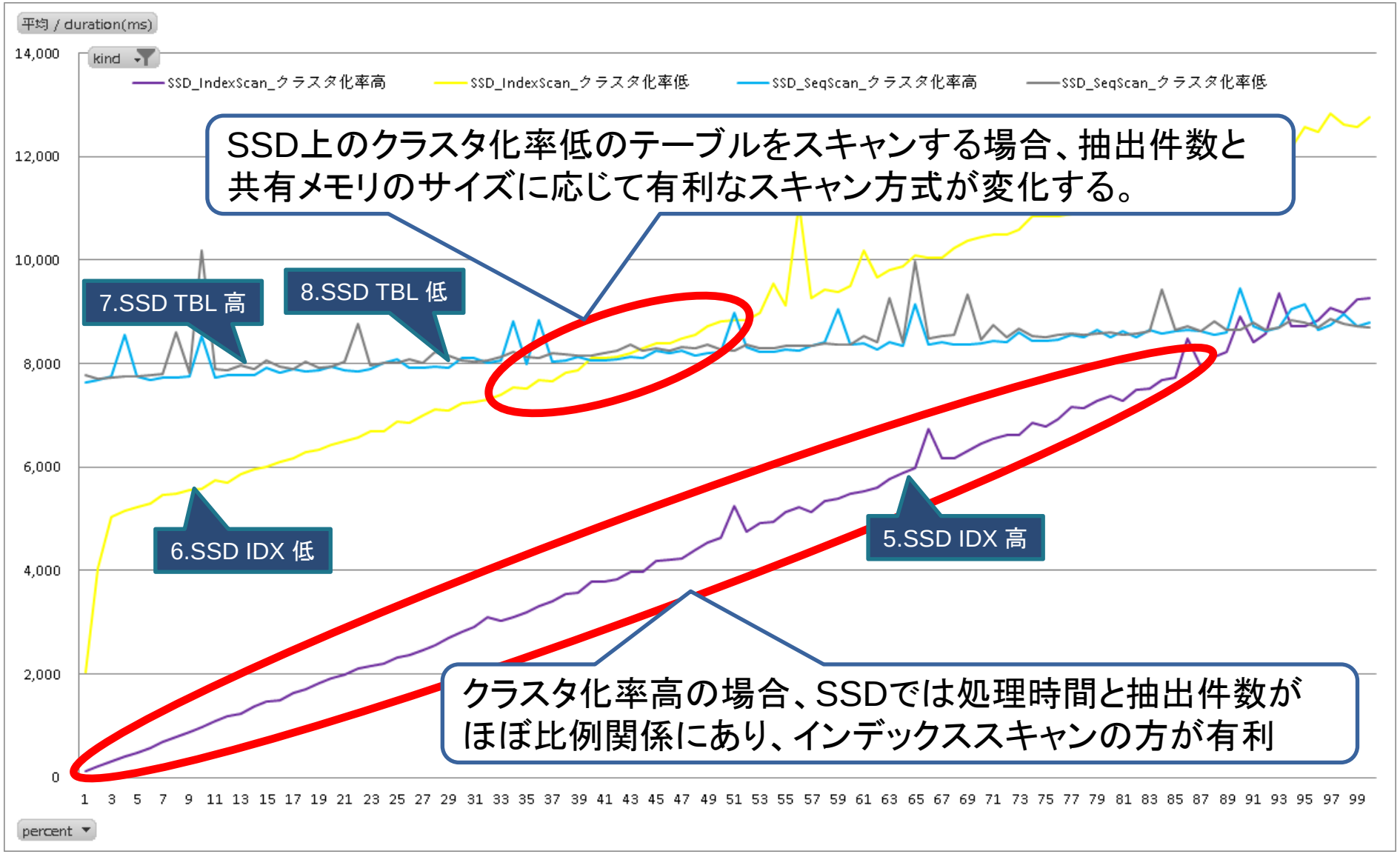


※ 横軸はデータの取得割合（1～100%）、縦軸は処理時間 (ms) を示す

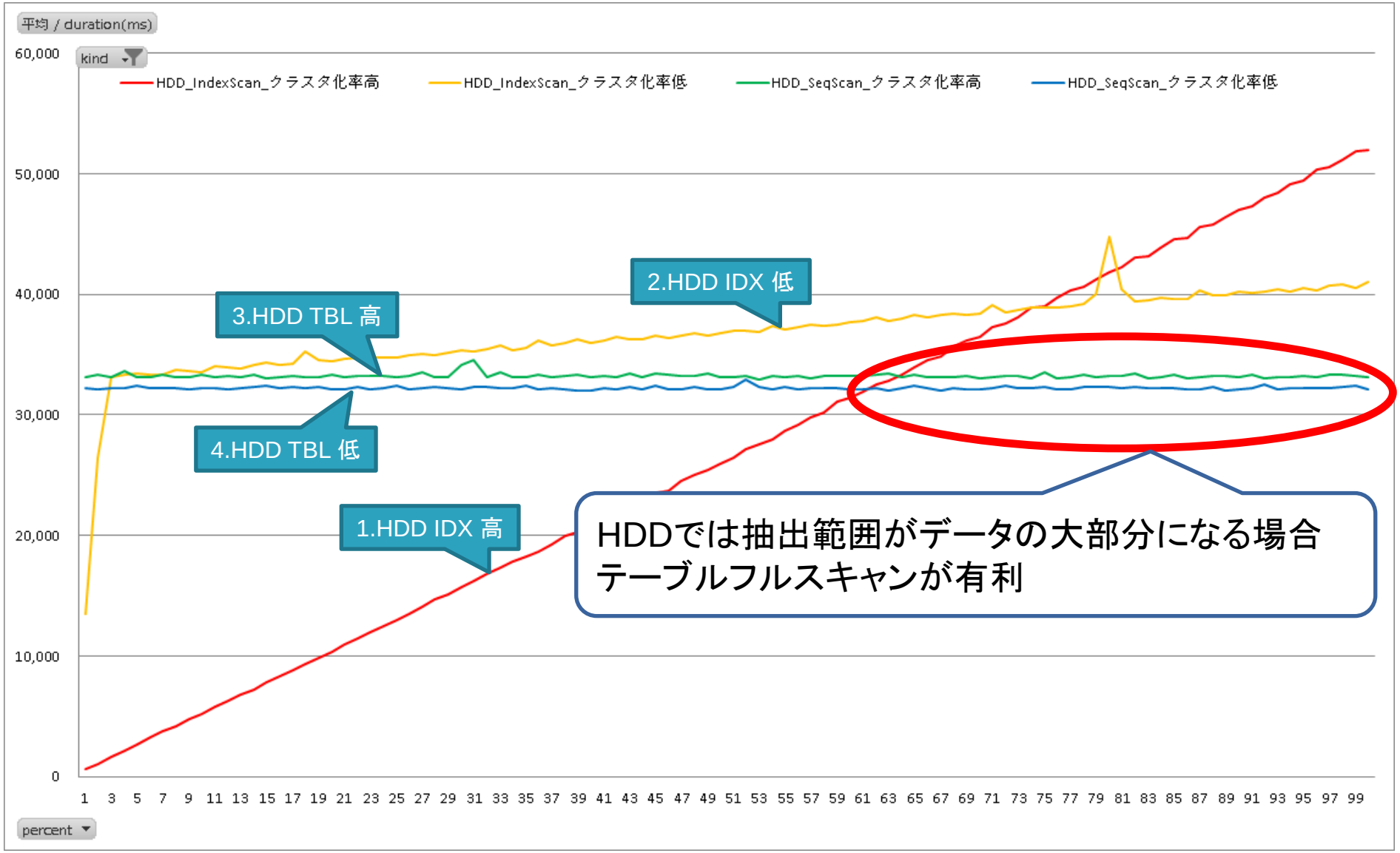
2.6.少量データ抽出ケースの傾向



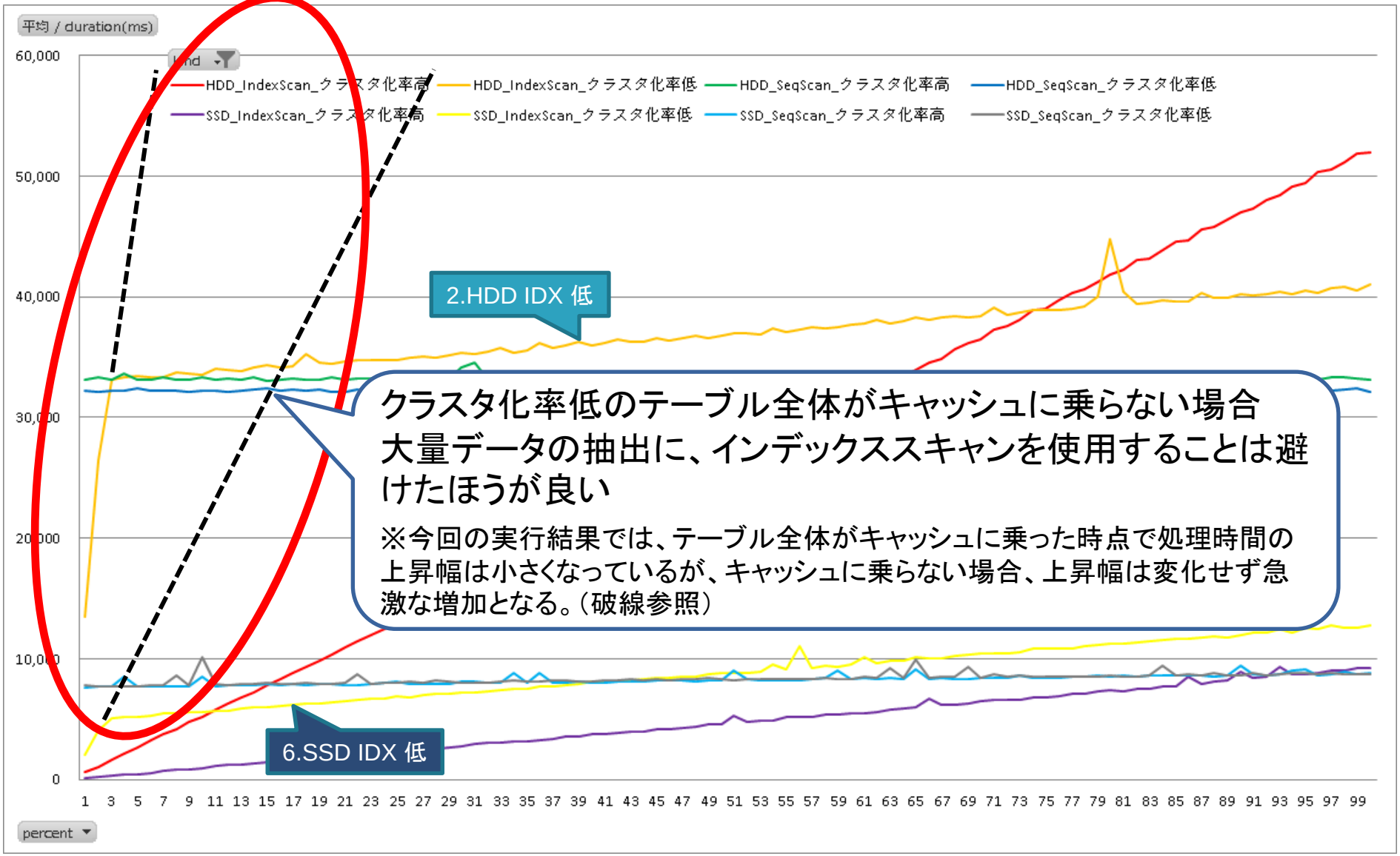
2.7.SSDでのクラスタ化率による傾向



2.8.HDDでの大量データ抽出時の傾向



2.9.キャッシュが小さい場合の傾向



2.10.まとめ

- 抽出量が少ない場合はインデックススキャンが有利、多い場合はテーブルフルスキャンが有利

- インデックススキャンとテーブルフルスキャンの処理時間が逆転する抽出量は、テーブル内のデータの並び方により変化する(HDD:3～62%、SSD:40～93%)

- SSDはHDDより全体的に処理時間が短くなるが、抽出量やスキャン方式による処理時間変化の傾向はHDDと大きく変わらない

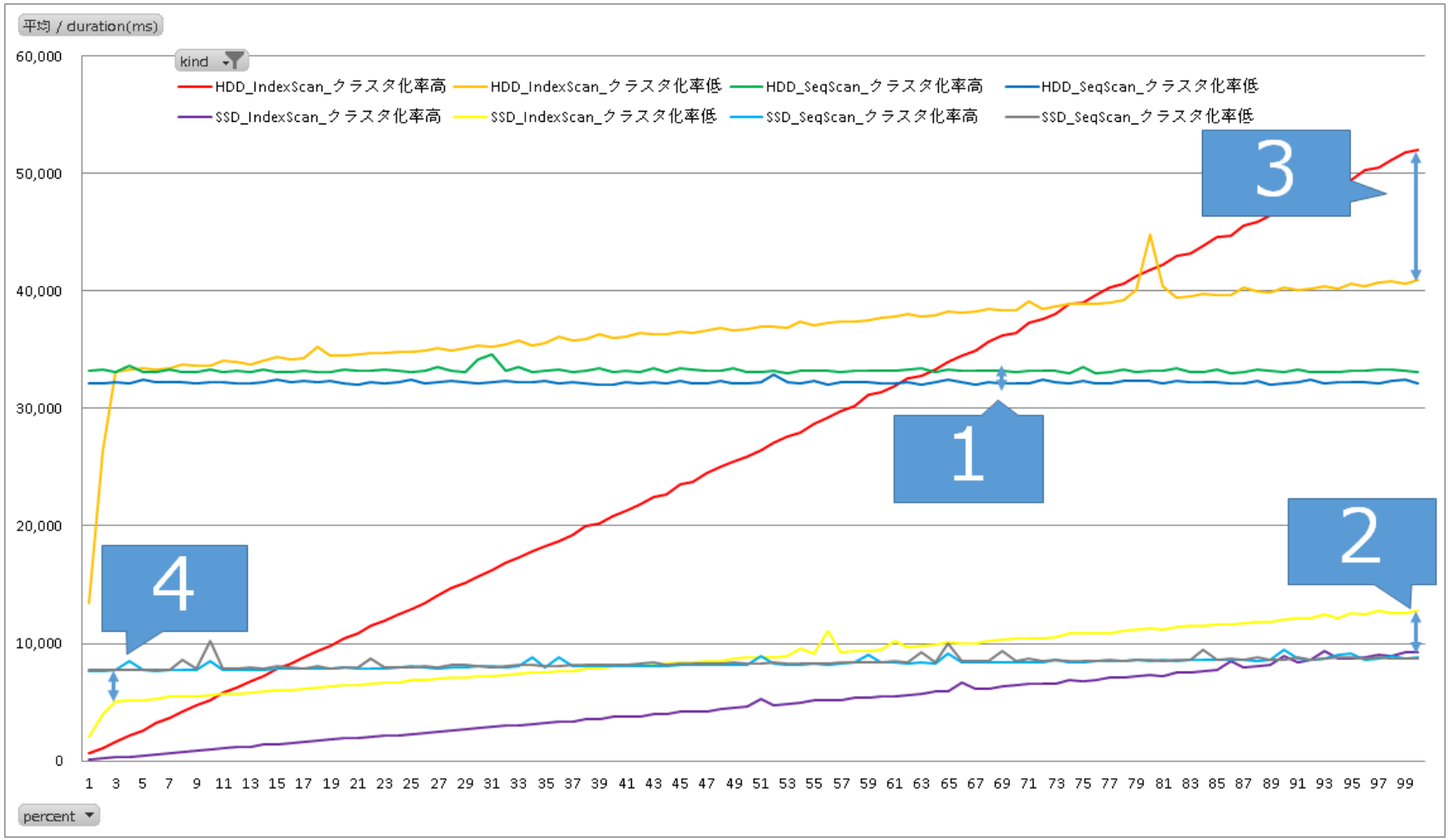
PostgreSQL Enterprise Consortium

POSTGRESQL ENTERPRISE CONSORTIUM

補足資料.　想定と異なる検証結果と原因分析

検証結果について、想定と異なる箇所が見られたため、その原因について調査を実施。
詳細はWebの成果物を参照。

補足1.想定外の傾向箇所



※ 横軸はデータの取得割合（1～100%）、縦軸は処理時間(ms)を示す

補足2.傾向に対する考察1

1. HDD のテーブルフルスキャン2種類でクラスタ化率低の方がわずかに速い

想定

・参照するデータ量が同じであるため、処理時間が同じ

推測

・各データファイルのファイルサイスは同一（1.95GB）だが、DDで読出し時間を計測すると0.7秒差がある（５回計測の平均値）
そのため、ファイルのフラグメンテーションによる差と考えられる

確認

・デフラグ後再度計測した結果ほぼ同じ処理時間となることを確認

補足3.傾向に対する考察2

2. SSD のインデックススキャン2種類で 100% のデータ読出し時、クラスタ化率低の方が遅い

想定

・100%では参照するデータ量が同じであるため、処理時間が同じ

推測

・SSDからの総読出し時間は同じだが、クラスタ化率高と比較しクラスタ化率低にてページイン/ページアウトの発生頻度が高い

確認

・パフォーマンス計測ツール(perf)にて CPU キャッシュ状況を計測しクラスタ化率低の方が頻繁な書き換えが発生していることを確認

補足4.傾向に対する考察3

3. HDD のインデックススキャン2種類で 100% のデータ読出し時、クラスタ化率低の方が速い

想定

・SSDのインデックススキャン2種類と傾向が同じ

推測

・インデックススキャンの際、クラスタ化率低と比較してクラスタ化率高はヘッドシークの発生頻度が高い

確認

・インデックスをテーブルとは物理的に別のディスク(SSD)に配置し再度計測した結果、SSDと同じ傾向となったことを確認

補足5.傾向に対する考察4

4. SSD の データ読み出しが 2.5% の時、クラスタ化率高の インデクススキャンの方がテーブルフルスキャンよりも速い

想定

・ファイルからの読込ブロック数は同じであり、処理時間も同じ

推測

・SSDの場合、PostgreSQL のレコード処理中は先読み実行(Read Ahead)が並列に動けないため、SSDの読み込みとレコード処理が直列化される
・HDDの場合は PostgreSQL のレコード処理と先読み処理にが並列して行われるため、全体の処理時間はディスクからの読込時間となり、処理時間はほぼ同じになる

確認

・力不足により確認方法が分からず、今後の課題とする
（情報提供をお待ちしております）

参考情報: 実行環境

- OS: CentOS Linux release 7.7.1908
- HHDとSSDの読み取り性能:
  - HDD:　60.5MB/秒
  - SSD: 432.0MB/秒
  - 確認コマンド
    dd if=<HDDまたはSSDに作成したファイル>
    of=/dev/null bs=1M count=1024
- PotgreSQL: 11.6

参考情報: テーブル構成

```
CREATE TABLE
    test
    (   col01       char(30),
            -- 主キー。
            -- 1 から 10,000,000 の数値を左ゼロ埋めした値。
        col02       char(139),
            -- ダミーデータ
        CONSTRAINT test_pk PRIMARY KEY( col01 )
    )
```

参考情報:インデックススキャン時のクエリ

```
SELECT
    *
FROM
    test
WHERE
    col01    >=   '00000000000000000000000000000001'
AND col01    <=   '00000000000000000000000250000'
    -- 1%から100%の割合で変化させる
```

参考情報:テーブルフルスキャン時のクエリ

```
SELECT
    *
FROM
    test
WHERE
    col01 || '_' >=  '00000000000000000000000000001' || '_'
AND col01 || '_' <=  '00000000000000000000000250000' || '_'
    -- 1%から100%の割合で変化させる
    -- カラムに文字列連結することでインデックススキャンを
       選択させない
```

参考情報: PostgreSQL 実行計画関係の設定

enable_bitmapscan	off
enable_gathermerge	on
enable_hashagg	off
enable_hashjoin	off
enable_indexonlyscan	off
enable_indexscan	on
enable_material	off
enable_mergejoin	off
enable_nestloop	on
enable_parallel_append	off
enable_parallel_hash	off
enable_partition_pruning	off
enable_partitionwise_aggregate	off
enable_partitionwise_join	off
enable_seqscan	**off**
enable_sort	on
enable_tidscan	on

※ 大量データ参照時でもインデックススキャンが選択されるよう enable_seqscan を off に設定

参考文献

- 文献1. PostgreSQLのインデックス・チューニング by Tomonari Katsumata

  https://www.slideshare.net/InsightTechnology/dbts-osaka-2014-b23-postgresql-tomonari-katsumata

- 文献2. PostgreSQL SQLチューニング入門　実践編

  https://www.slideshare.net/satoshiyamada71697/postgresql-sql