MySQLデータベースにおける
Fusion-io社 ioDrive使用時の優位性について

2012/07/23@GMO yours

- MySQLデータベースにおける
  Fusion-io社 ioDrive使用時の優位性について

- 事例紹介
  ～Too many connections

- MySQLデータベースにおけるパフォーマンス向上の手段として、ストレージ媒体をハードディスクドライブ（以下HDD）からFusion-io株式会社 ioDrive（以下ioDrive）に変更する選択肢に関して、株式会社スマートスタイルがその有効性を検証した過程を記載したものです。

- 本書に記載された測定記録は、同様の構成を取った場合でも同じ結果を保証するものではありません。

- 本書の執筆に際し全面的なご協力をいただきましたGMOインターネット株式会社様、Fusion-io株式会社様に、この場を借りて御礼申し上げます。

測定緒元

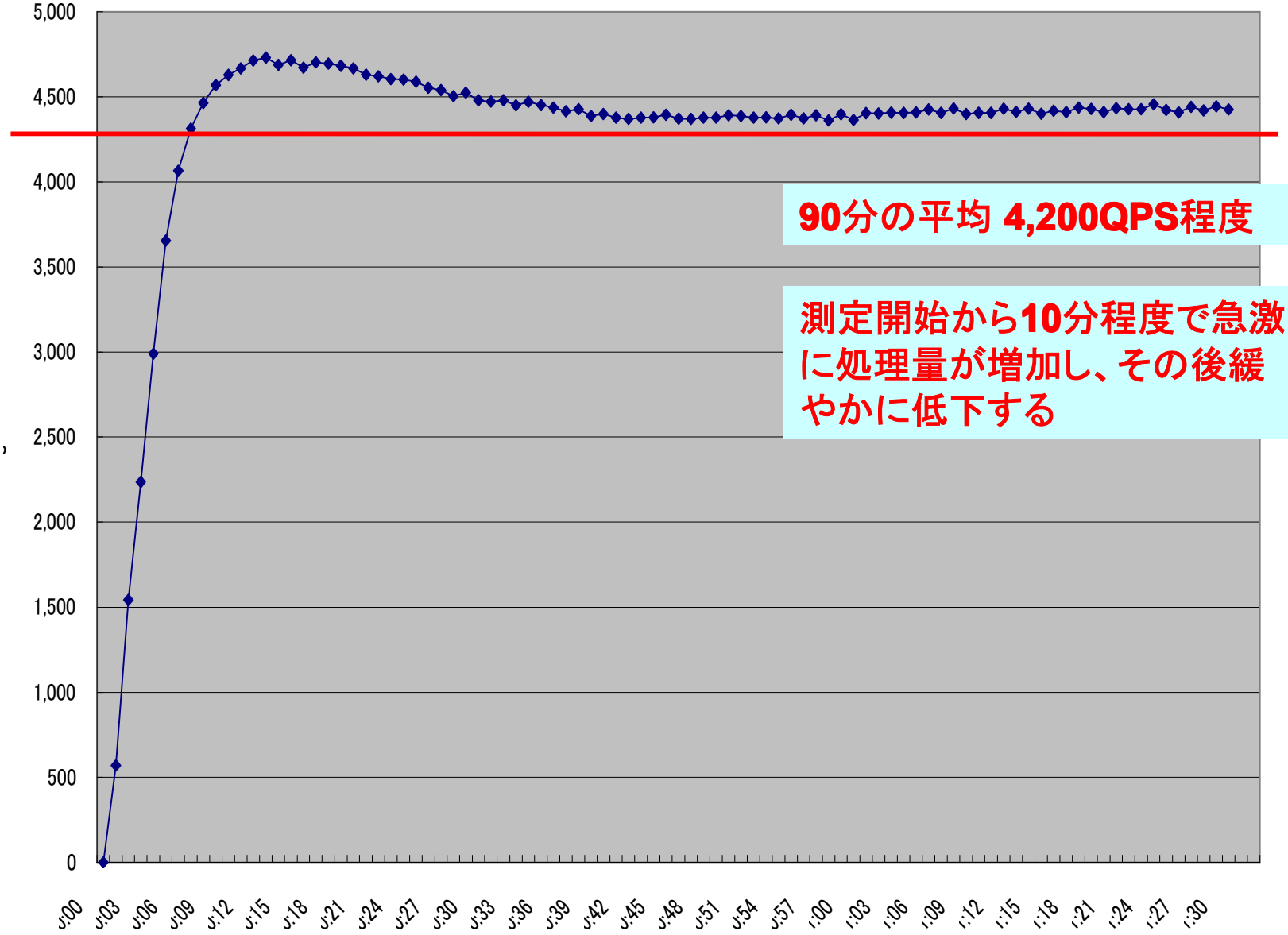
SmartStyle

- CPU　Intel Xeon CPU E5620 * 2
- RAM　64GB
- OS　Red Hat Enterprize Linux 5(2.6.18)
- MySQL 5.5.25-log Community Server (GPL)
- HDD　SAS 146GB　6台(Hardware RAID1+0)
- ioDrive　ioDrive Duo 320SLC 2台(Software RAID)
- 負荷テストツールpercona-tools tpcc-mysql

・負荷計測の直前にOSを再起動し、バッファ・キャッシュの内容をクリアする
・データベーステーブル数 9(tpcc-mysqlによる作成)
・負荷クライアントからの同時実行スレッド20(tpcc-mysqlによる実行)
・データベース起動直後から計測を行う
・tpcc-mysqlの算出するtpmCの値ではなく毎分のCom_queryの増分を計測し
　QPS(Query Per Second)の推移を記録する
・測定は90分間を3回行い、値は記録された数値の平均を採用する
・メモリ関連のMySQLパラメータはHDD使用時、ioDrive使用時で同じものを使用するが、I/O関連のパラメータに関してはそれぞれの特
　性が出る様チューニングを実施する

まずは比較元としてHDD使用時の測定結果

HDD + Innodb_buffer_pool_size=256M



H/Wのスケールアップとして、

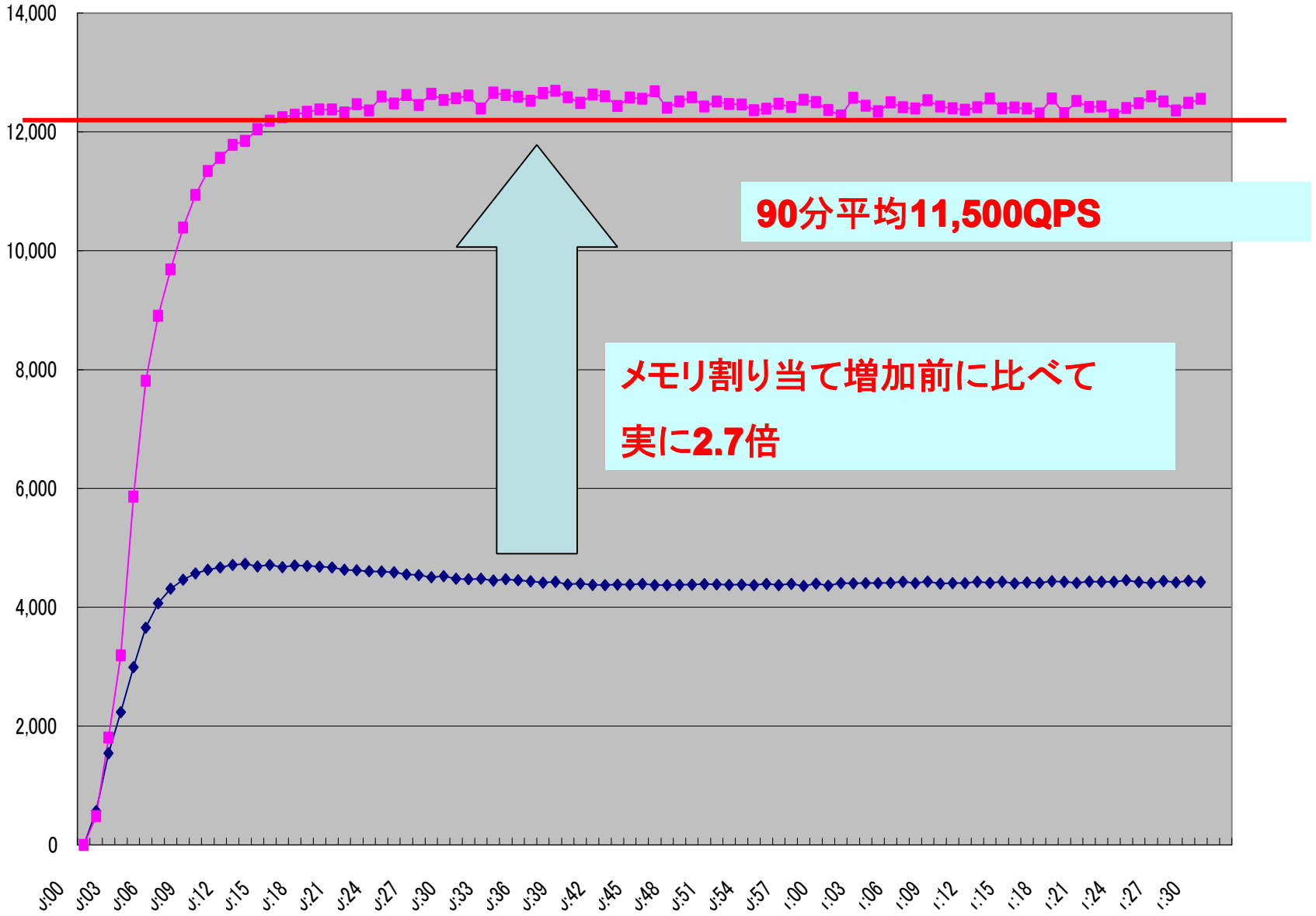
1) メモリの増設（今回は割り当て量増加で代替）

2) ioDrive化

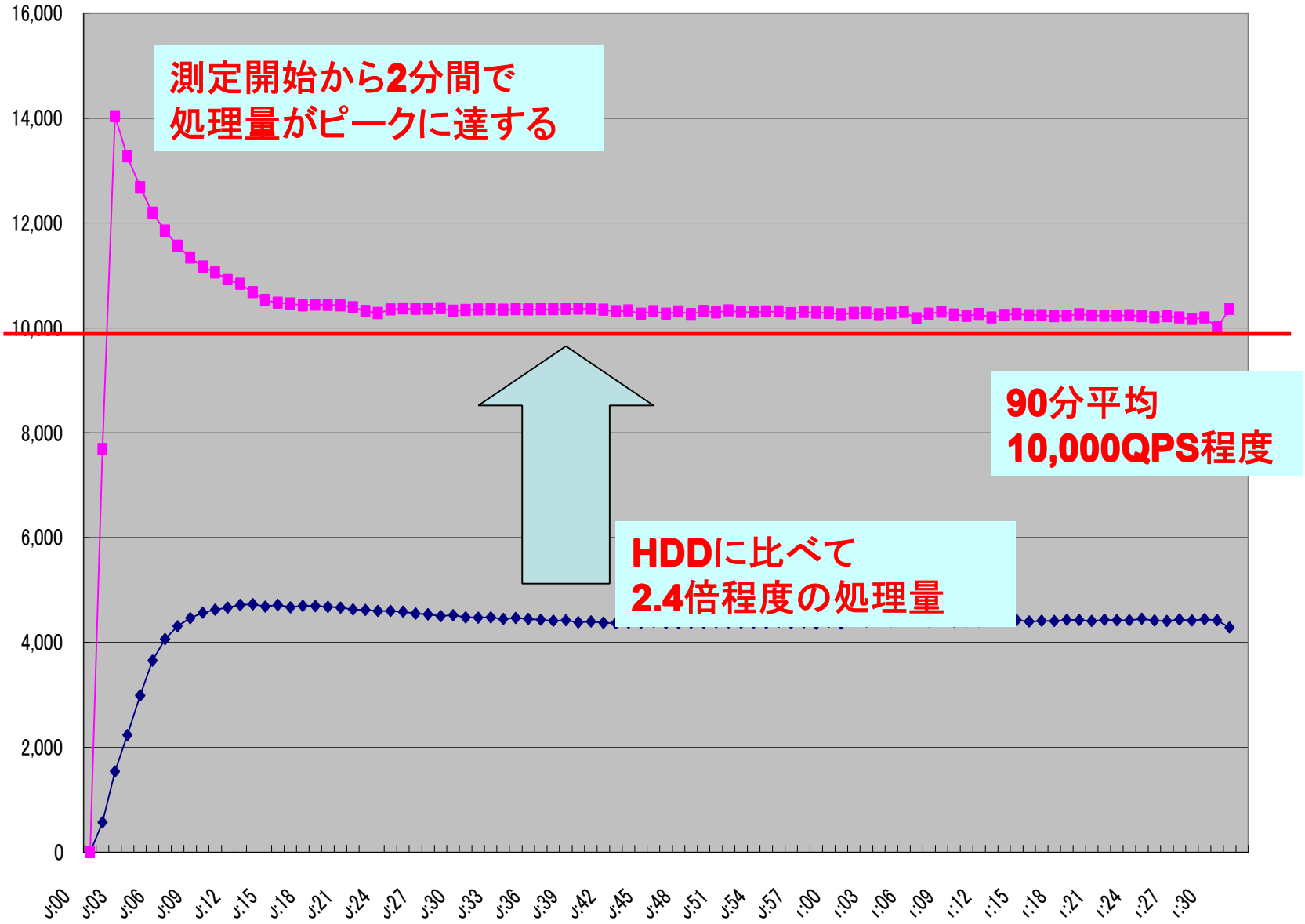
のどちらがより有効かを比較します。

HDD + Innodb_buffer_pool_size=40G

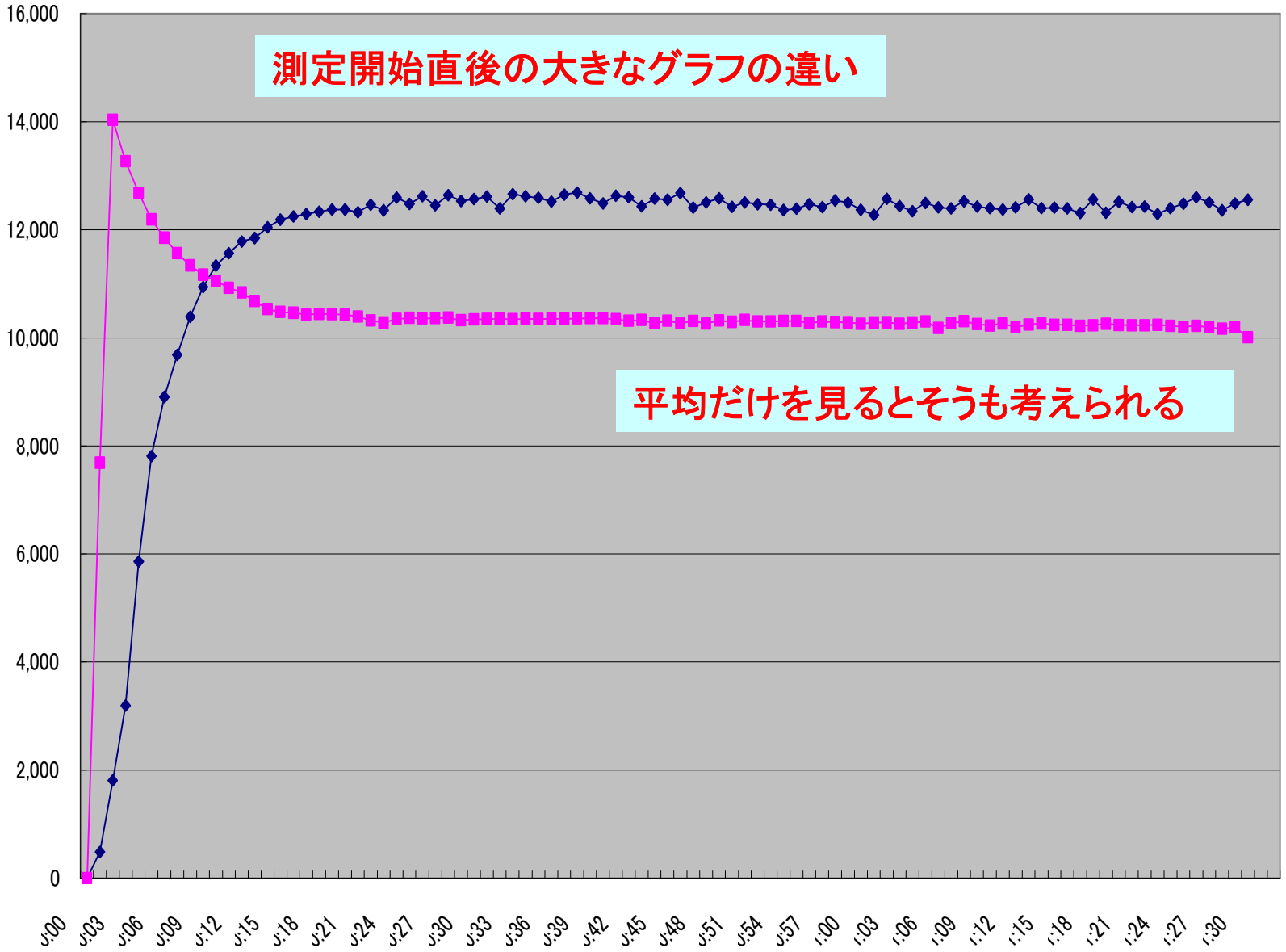
SmartStyle



ioDrive + innodb_buffer_pool_size=256M



ioDrive化よりもメモリ増設が有効なのか？

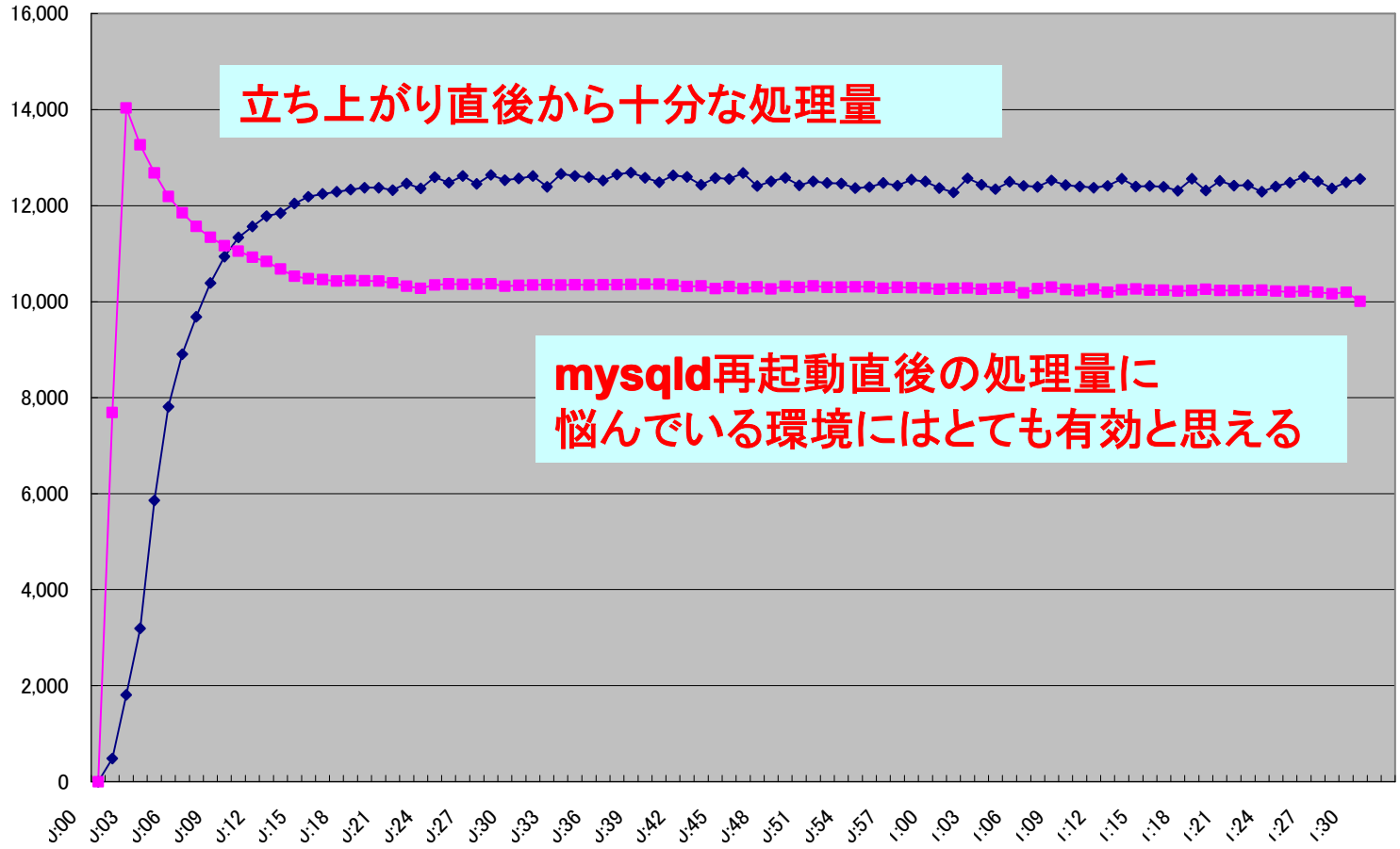


起動直後の優位性

SmartStyle

- メンテナンスやH/Wダウンで MySQL停止
- 起動後しばらくの間レスポンスが悪く、
  アプリのタイムアウトや
  Too many connection発生
- mysqld起動直後はバッファが空なので、
  バッファが暖まるまでは
  ストレージから読み込むしかない。
- 起動直後はかなり処理量が低下する

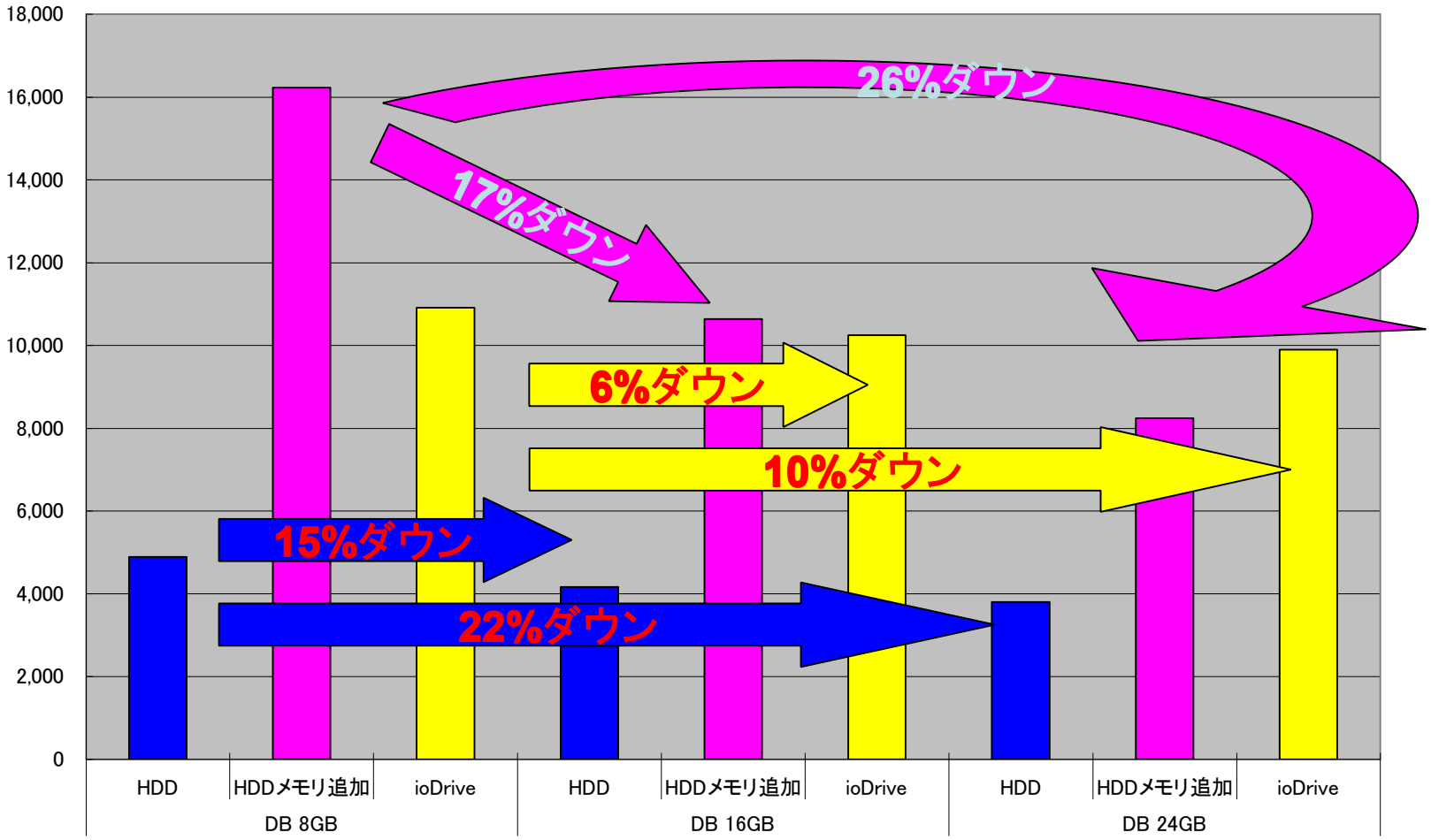
起動直後の優位性



- データベースが肥大化していくほど、
  処理量は低下していく

- HDDとioDriveで
  処理量の低下具合に差があるのかどうかを計測

データベースサイズごとの平均処理量



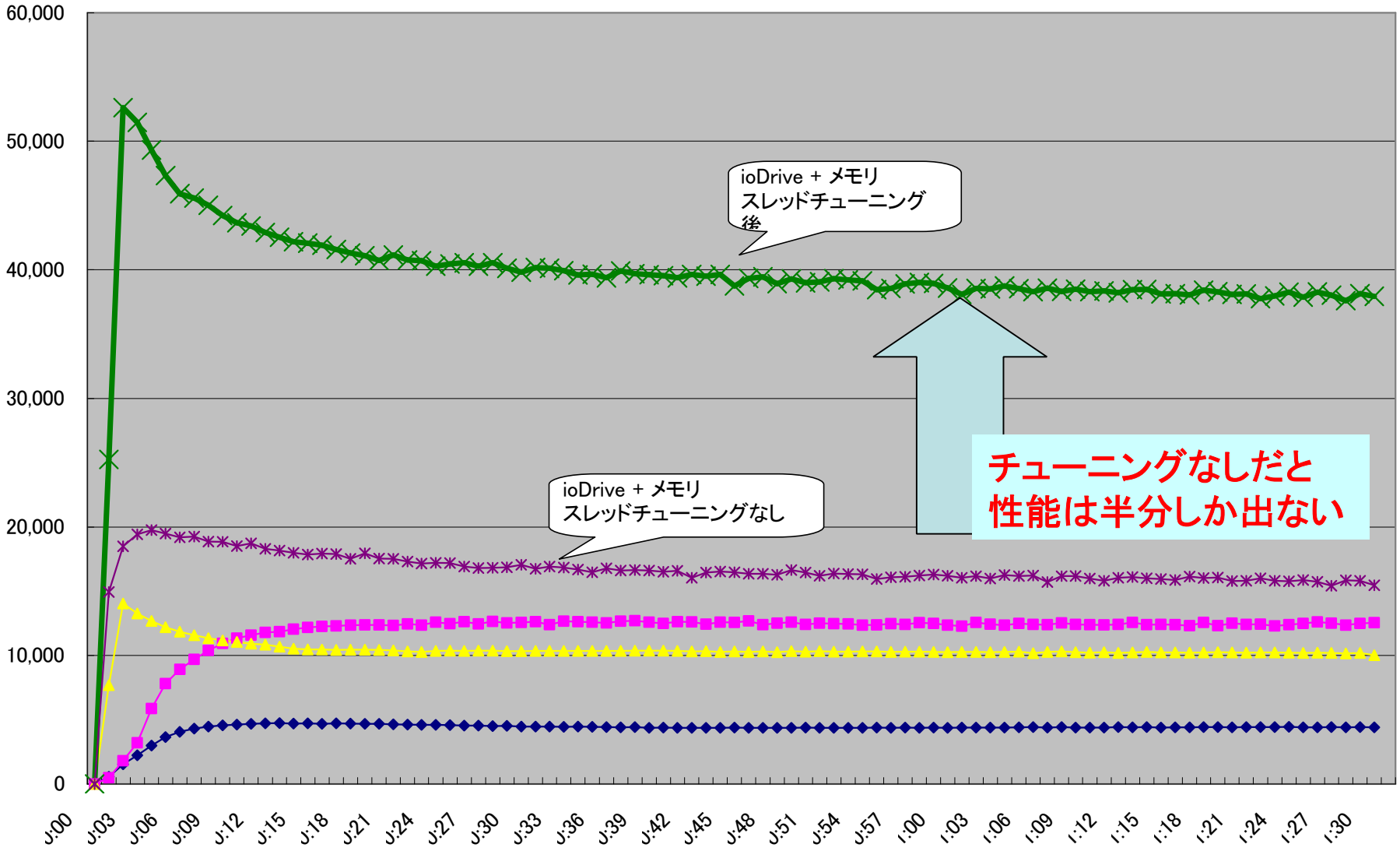
- HDDを使っている場合は
  DBサイズが増えるにつれ
  かなりの速度で処理量が減少していく

- ioDriveを使用しているケースは
  DBサイズの増加と比較して
  処理量減少が圧倒的に穏やか

- DBサイズ16GBのケースでメモリの増強と比較して
  ほぼ同程度、DBサイズ24GBのケースでは
  メモリの増強よりも処理量向上率は高い

ioDrive + innodb_buffer_pool_size=40G

SmartStyle



こんな方々に特にioDriveが有効だと思います。

- メンテナンス明けのレスポンス低下
- 物理的にこれ以上メモリが増設できない
- データベースサイズが大きくなりすぎて
  導入当初に比べて性能劣化が大きい
- アプリケーションの修正に時間が割けない
- とにかく速さを追求したい

最近ソーシャル関連のお客様から
お問い合わせのあった事例として

・アクセス数の増加に伴いWEBサーバを増設、
　MySQLへの接続数が増えた
・イベント開催に伴いアクセスがバーストし、
　MySQLへの接続数が増えた

結果、Too many connectionsというエラーが発生

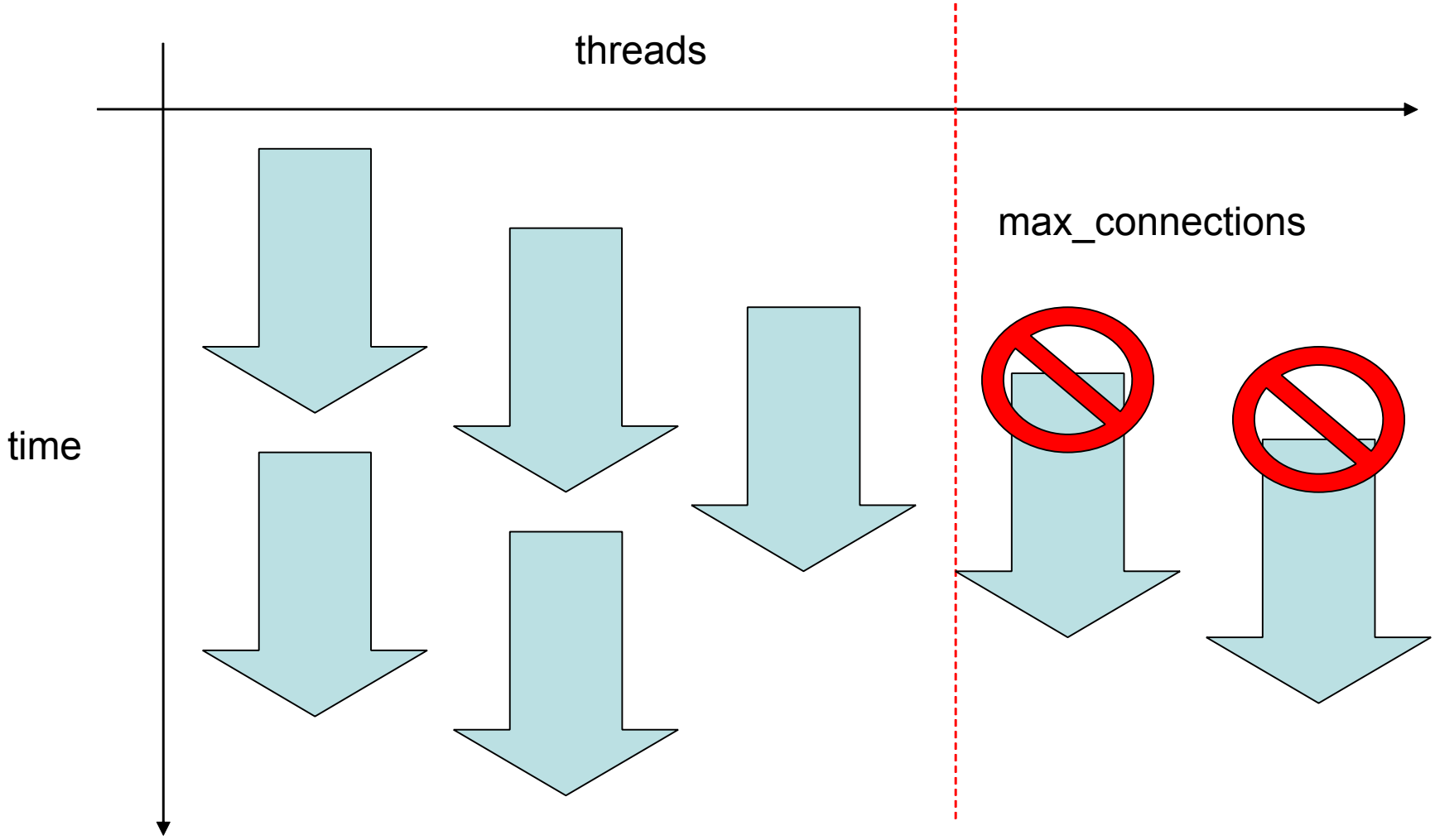
- Too many connectionsとは

  アプリケーションからMySQLに接続できる数はmy.cnfで設定されている(max_connections)

  max_connections以上の接続をしようとすると出るエラー。

max_connectionsの値を増やすことで
エラーの回避は可能だが、
全体的なスループットは低下する傾向にある。
（アプリ側でリトライ処理を実装しなくて良いというメリットはある）

Too many connectionsのメカニズム



max_connectionsを増やすと

SmartStyle

- スレッドバッファ分メモリ使う

⇒検証環境でstraceを追ってmmap関数を調べたところ、
1スレッド生成あたり64MB程度メモリを確保している。

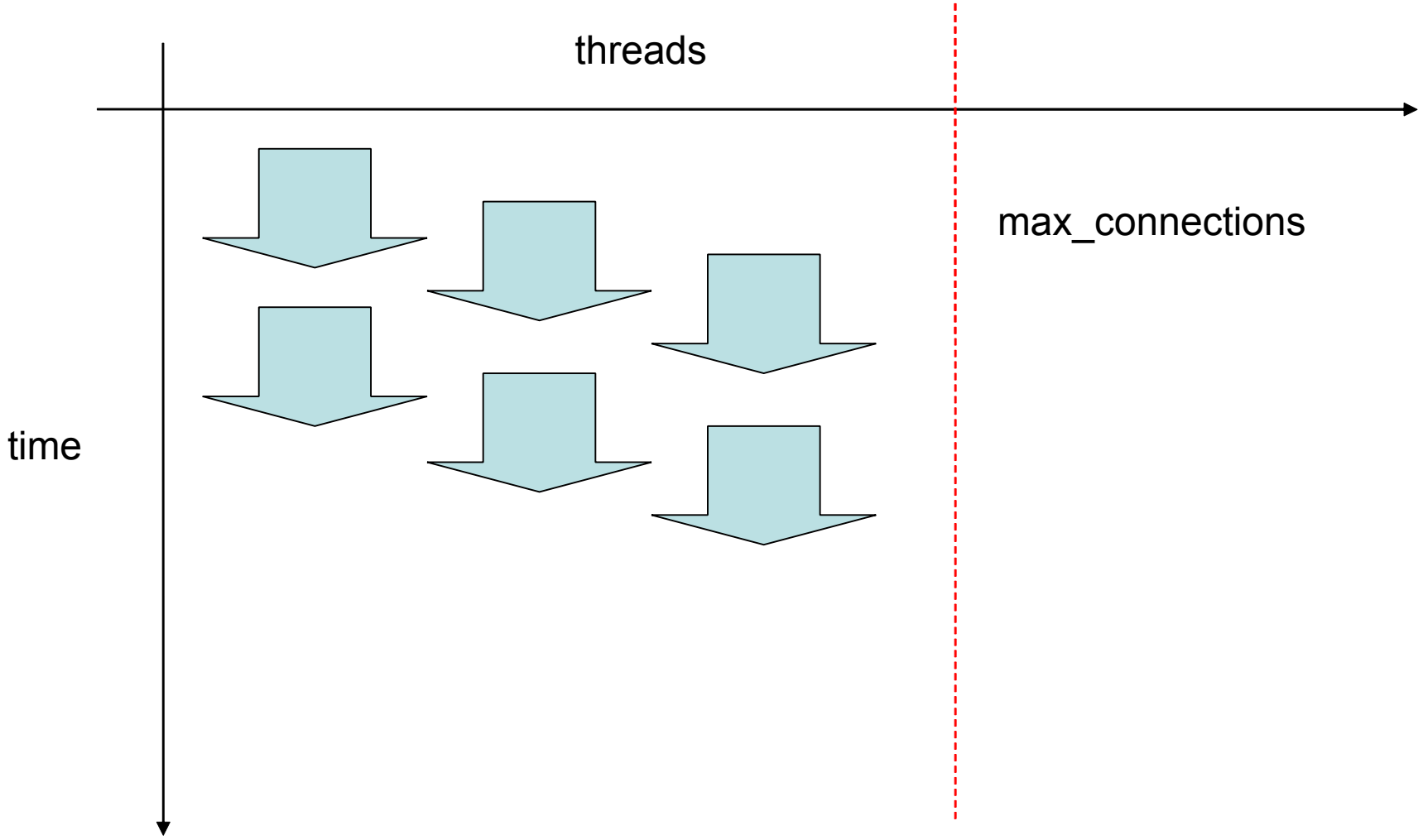
⇒スレッドが100本生成されるなら、6GBにもなる。

- スレッド生成時にオーバーヘッドがかかる

⇒検証環境でstraceを追った時の待機状態になるまでの時間。
スレッドキャッシュなし .. 50ms～150ms程度
スレッドキャッシュあり .. 15ms～35ms程度

これらがもともとのクエリの速度を圧迫する

理想的な解決法



実験

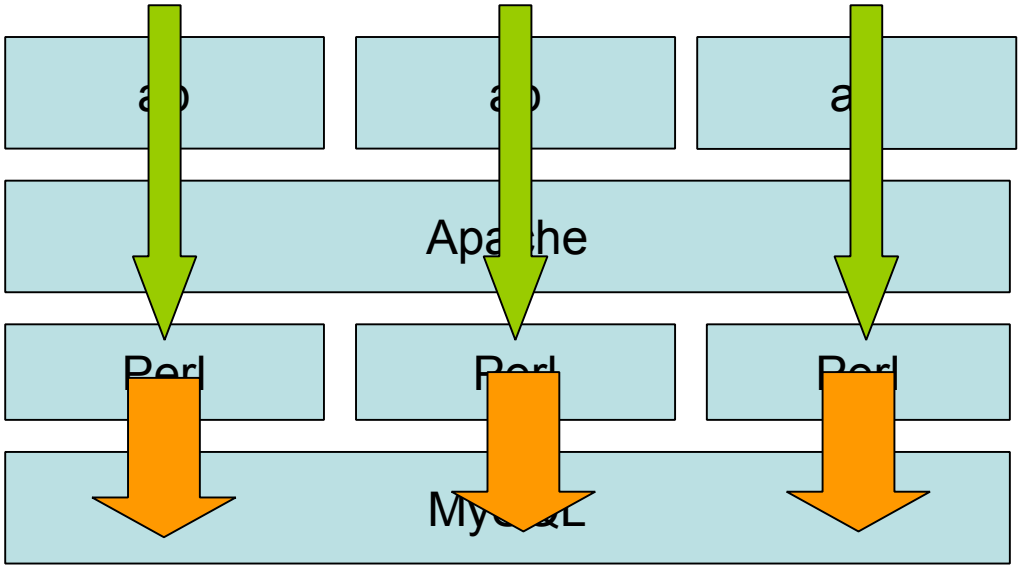
SmartStyle

ローカルホストにLinux,Apache,MySQL,Perlの環境を作成。

abクライアントからApache越しにPerlスクリプトを叩く。

PerlスクリプトはMySQL10多重アクセスをして結果を返す。

そのターンアラウンドタイムを計測。



abは3多重で起動

Perlは10多重で
MySQLへアクセス

結果

SmartStyle

	Too many connections発生回数	平均レスポンスタイム
クエリチューニングなし max_connections=10	19回/30回	6.2秒
クエリチューニングなし max_connections=151	0回/30回	17秒
クエリチューニングあり max_connections=10	0回/30回	0.5秒

まとめ

SmartStyle

- max_connectionsを増やして
  レスポンスが向上するケースは、
  メモリ,CPU,NIC,I/Oいずれも余裕があり、
  connection数だけがボトルネックになっている
  ケースのみ

- max_connectionsを増やすことで、
  全体のスループットが悪くなるケースもままある

- アプリ側でリトライ処理を実装しなくて良いという
  メリットはあるがスループット低下のリスクも高い

- 全体的なスループットを保ちつつ状況を改善する為には
  スキーマやSQLのチューニングが必須