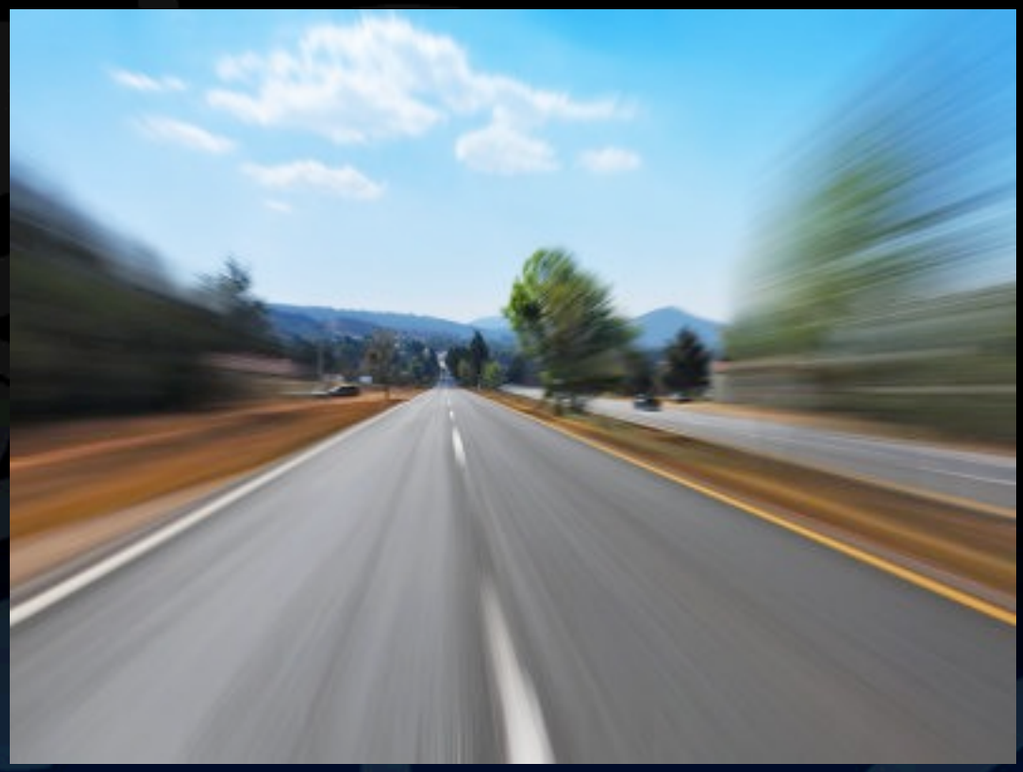

最適化

最適化

シニア・ソフトウェア・エヴァンジェリスト
ティボウ・アルギエール THIBAUD ARGUILLÈRE

about

命題

running faster

データベースを速くすること

about

命題

- ~~開発を早くすること~~
- ~~DB設計を改良すること~~
- ~~保守に手のかからないコードを書くこと~~
- ~~v13の売り込み~~

the best advice

最高のアドバイス

do not optimise

最適化をしない

the Pareto Principle

パレートの法則

80 20

faster, really?

もっと速く...ほんとうに?

- スピードは実際に必要とされている?

- スピードに不平を述べているのは誰?

最適化しない

easy optimisation

簡単にできる最適化

コンパイル

easy optimisation

簡単にできる最適化

ハードウェアをグレードアップ

easy optimisation

簡単にできる最適化

HDD vs SSD

Mac Pro 2008

- MacPro3,1
- 2x Quad-Core Intel Xeon
- 8 Cores
- 3.2 GHz
- 8 GB RAM
- WD Caviar 500 GB HDD

MacBook Pro 2006

- MacBookPro2,2
- Intel Core 2 Duo
- 2 Cores
- 2.33 GHz
- 2 GB RAM
- Intel X25-M 80 GB SSD

easy optimisation

簡単にできる最適化

Mac Pro 2008 vs MacBook Pro 2006

strict compare strings

バイト単位の文字列比較

RTFM

RTFM

RTFM

Read The Fxxxing Manualの略。

きちんとマニュアルを読め，という意味。元々は米空軍で使用された表現。

職場やインターネット掲示板など，使用する場面を誤れば，相手以上に怠惰あるいは無礼な返答となりかねないが，ここでは要するに，技術者が探している情報は，大概，マニュアルに記述されている，ということである。

triggers

トリガ

トリガ＝悪なのか?

普段は違う

しかし…

悪に転じることもある

miscellaneous optimisation

いろいろな最適化 #1

cost of defensive programming

防御的プログミングの対価

miscellaneous optimisation

いろいろな最適化 #2
%R コンパイラ指示子

```
For ($i;1;
   For ($j;1;$
      $ignore:=$arr
   End for
End for
```

最高40%速度アップ!
数十億回ループすれば,
の話ですが。

miscellaneous optimisation

いろいろな最適化 #3

cache function results

関数の値を取り分ける

cache function results

関数の値を取り分ける

```
theValue:=0
For ($i;1;$L_kMAX)
theValue:=theValue+Size of array(theArray)
End for
```

```
theValue:=0
size:=Size of array(theArray)
For ($i;1;$L_kMAX)
    theValue:=theValue+size
End for
```

x2
(インタプリタ)

x150
(コンパイル)

miscellaneous optimisation

いろいろな最適化 #4

inlining

インライン化

miscellaneous optimisation

いろいろな最適化 #5

cleanup your code

ごみ捨て

cleanup your code

ごみ捨て


```
// クライアントで10分毎に実行
QUERY([請求]; . . .) => 5,000件ほどの請求書
CREATE SET([顧客];"顧客請求書")
FIRST RECORD([請求])
While (Not(End selection([請求])))
  QUERY([顧客]; [顧客]ID= [請求] 顧客ID)
  ADD TO SET([顧客]; "顧客請求書")
  NEXT RECORD([請求])
End while
USE SET("顧客請求書")
```

cleanup your code

ごみ捨て

サーバーに送られるリクエストの数は?

```
// クライアントで10分置きに実行
QUERY([請求]; . . .)  => 5,000件ほどの請求書
CREATE SET([顧客];"顧客請求書")
FIRST RECORD([請求])
While (Not(End selection([請求])))
  QUERY([顧客]; [顧客]ID= [請求] 顧客ID)
  ADD TO SET([顧客]; "顧客請求書")
  NEXT RECORD([請求])
End while
USE SET("顧客請求書")
```

cleanup your code

ごみ捨て

サーバーに送られるリクエストの数は?

```
// クライアントで10分置きに実行
QUERY([請求]; . . .)  => 5,000件ほどの請求書
CREATE SET([顧客];"顧客請求書")
FIRST RECORD([請求])
While (Not(End selection([請求])))
  QUERY([顧客]; [顧客]ID= [請求] 顧客ID)
  ADD TO SET([顧客]; "顧客請求書")
  NEXT RECORD([請求])
End while
USE SET("顧客請求書")
```

5,000

cleanup your code

ごみ捨て

サーバーに送られるリクエストの数は?

```
// クライアントで10分置きに実行
QUERY([請求];...)  => 5,000件ほどの請求書
CREATE SET([顧客];"顧客請求書")
FIRST RECORD([請求])
While (Not(End selection([請求])))
  QUERY([顧客]; [顧客]ID= [請求] 顧客ID)
  ADD TO SET([顧客]; "顧客請求書")
  NEXT RECORD([請求])
End while
USE SET("顧客請求書")
```

5,000
+ 5,000
= 10,000

cleanup your code

ごみ捨て

サーバーに送られるリクエストの数は?

```
// クライアントで10分置きに実行
QUERY([請求]; . . .)  => 5,000件ほどの請求書
CREATE SET([顧客];"顧客請求書")
FIRST RECORD([請求])
While (Not(End selection([請求])))
  QUERY([顧客]; [顧客]ID= [請求] 顧客ID)
  ADD TO SET([顧客]; "顧客請求書")
  NEXT RECORD([請求])
End while
USE SET("顧客請求書")
```

cleanup your code

ごみ捨て

サーバーに送られるリクエストの数は?

```
// クライアントで10分置きに実行
QUERY([請求]; . . .)  => 5,000件ほどの請求書
CREATE SET([顧客];"顧客請求書")
FIRST RECORD([請求])
While (Not(End selection([請求])))
  QUERY([顧客]; [顧客]ID= [請求]顧客ID)
  ADD TO SET([顧客]; "顧客請求書")
  NEXT RECORD([請求])
End while
USE SET("顧客請求書")
```

cleanup your code

ごみ捨て

サーバーに送られるリクエストの数は?

```
// クライアントで10分置きに実行
QUERY([請求];...)  => 5,000件ほどの請求書
RELATE ONE SELECTION([請求];[顧客])
```

miscellaneous optimisation

いろいろな最適化 #6

unicode

Unicodeモード

miscellaneous optimisation

いろいろな最適化 #6

バブルソート	インタプリタ	コンパイル
2004	19秒	8.5秒
SJISモード	19秒	8秒
Unicodeモード	1.9秒 (10x 高速)	0.11秒 (70x 高速)

data and server

データとサーバー

- 無意味なリクエスト
  - FIRST RECORD? LOAD RECORD? ほんとうに必要?
- セットと命名セレクションのスコープ
- プリエムプティブになりましょう!
- インデックスが遅さの原因になることもある
- キャッシュを理解する