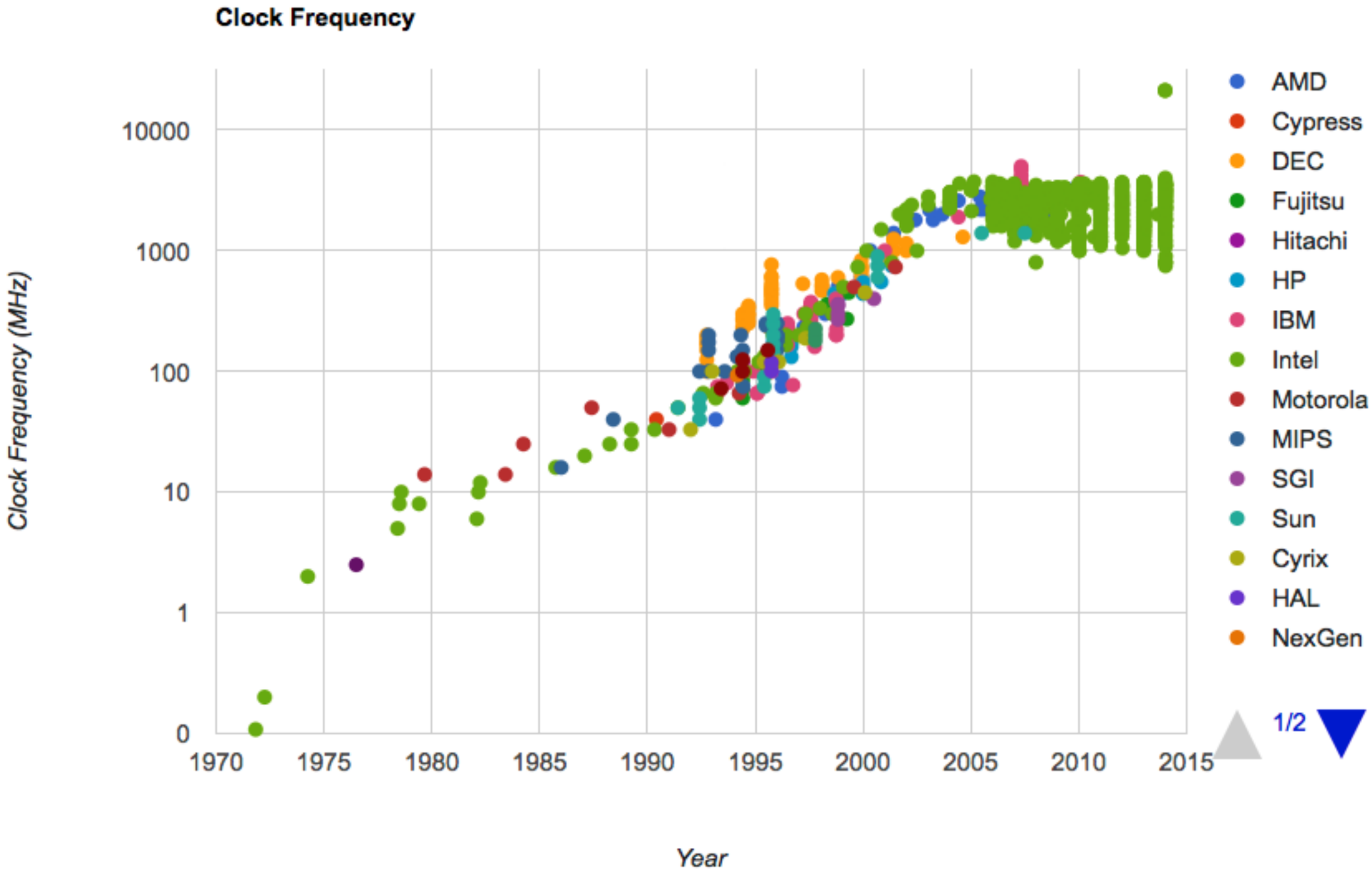
2015/08/25

AS.Parallel

Piyomaru Software
Takaaki Naganoya

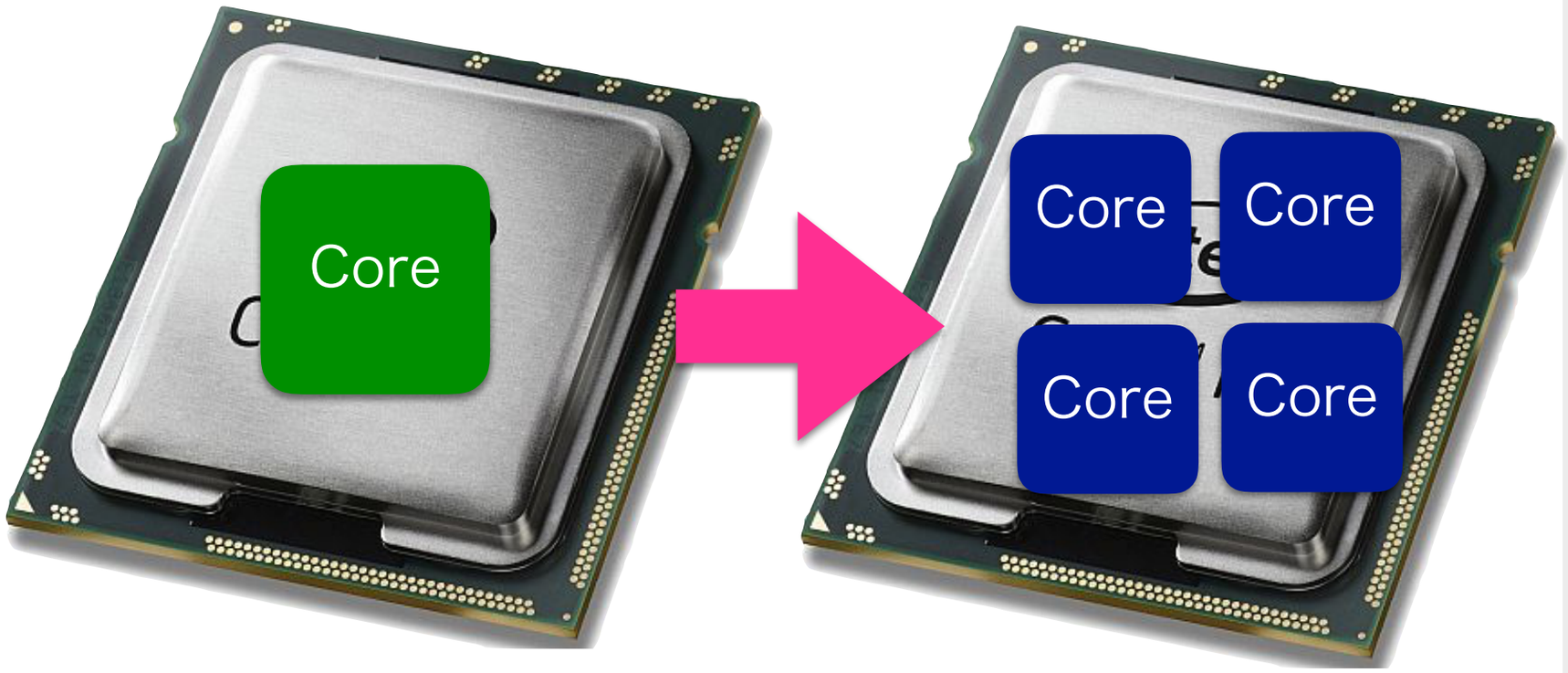
CPUコアの動作クロックの向上が鈍化

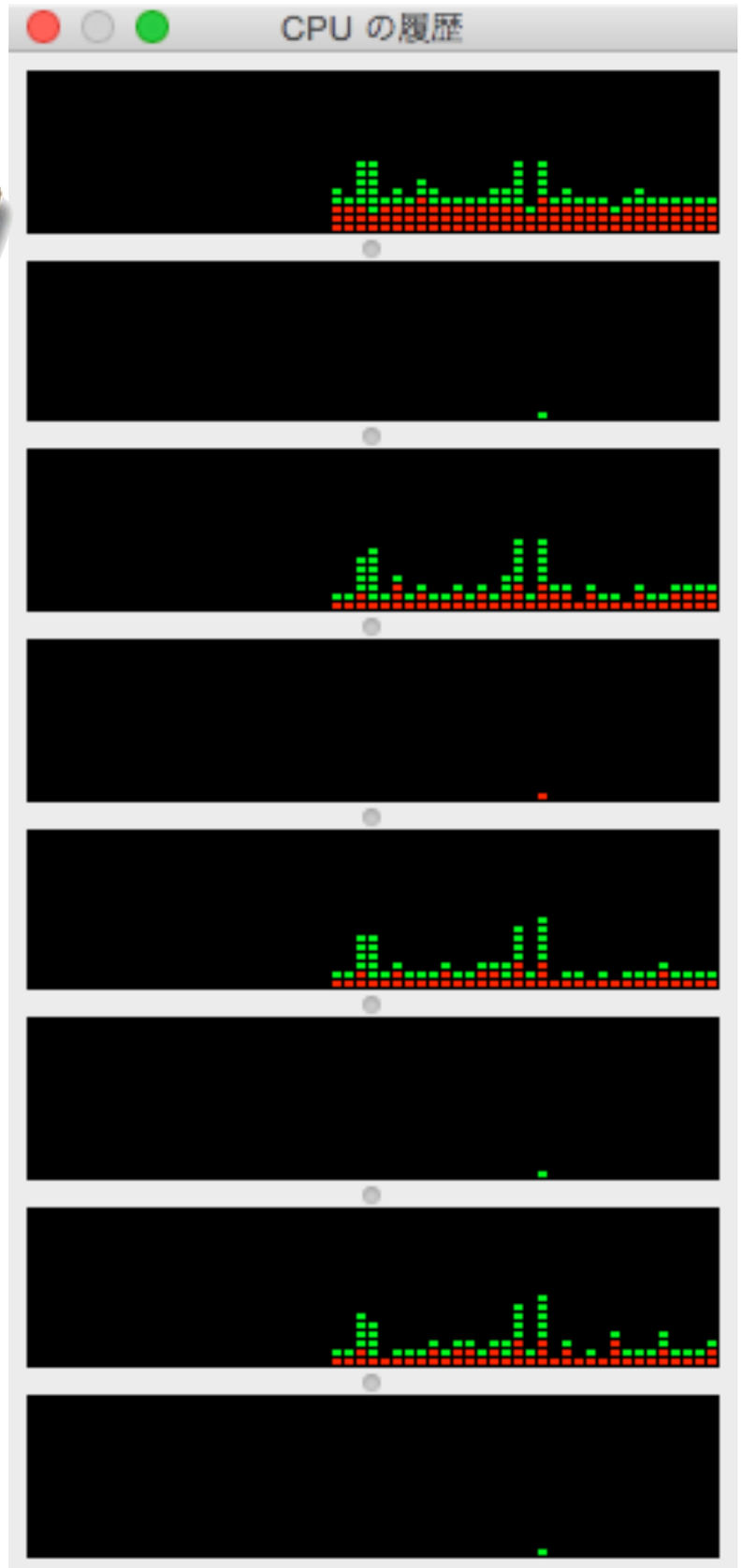
西暦2000年以降、CPUの動作クロック周波数の向上が鈍化



動作クロック周波数至上路線（Pentium 4あたり）から、実処理性能の向上へと（Dothan, Yonah=CoreDuo）路線が切り替わった

いまやメニーコアCPU時代



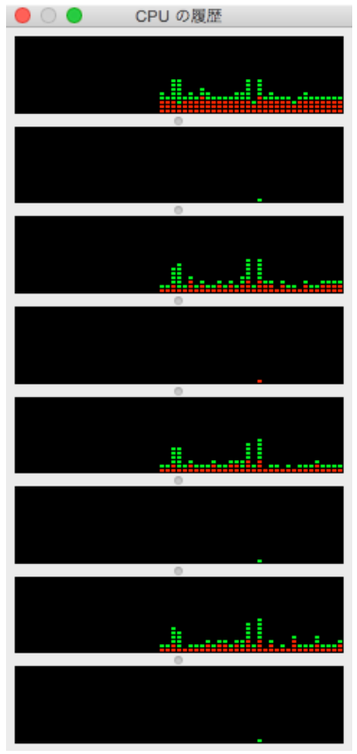


CPU１コアあたりの演算性能の向上が頭打ち

→　複数コアを搭載する方向に進化

デュアルコア・４スレッドが最低ライン。クアッドコア・８スレッドも珍しくない！

メニーコアを活用するにはソフト側の対応が必要



シングルスレッド

Software

Software

マルチスレッド対応アプリ

OSレベルでも、**Ground Central Dispatch**といった並列化のための仕組みもあり、AppleScriptでは「処理系そのもの」が複数スレッドで並列実行され、処理速度が大幅に向上しているものの、その上で動くScriptが並列化されてはいない

AppleScriptの並列処理対応

処理系がScriptを複数スレッドを用いて実行、処理速度向上

AppleScript処理系がスレッドセーフ（複数実行対応）に

AppleScriptをメニューから複数同時実行可能に

OSのバージョンが上がるたびに処理系のスレッド数が増加（マルチスレッド対応強化？）

→　ただし、複数CPUコアをAppleScript自体から利用できるようにはなっていない

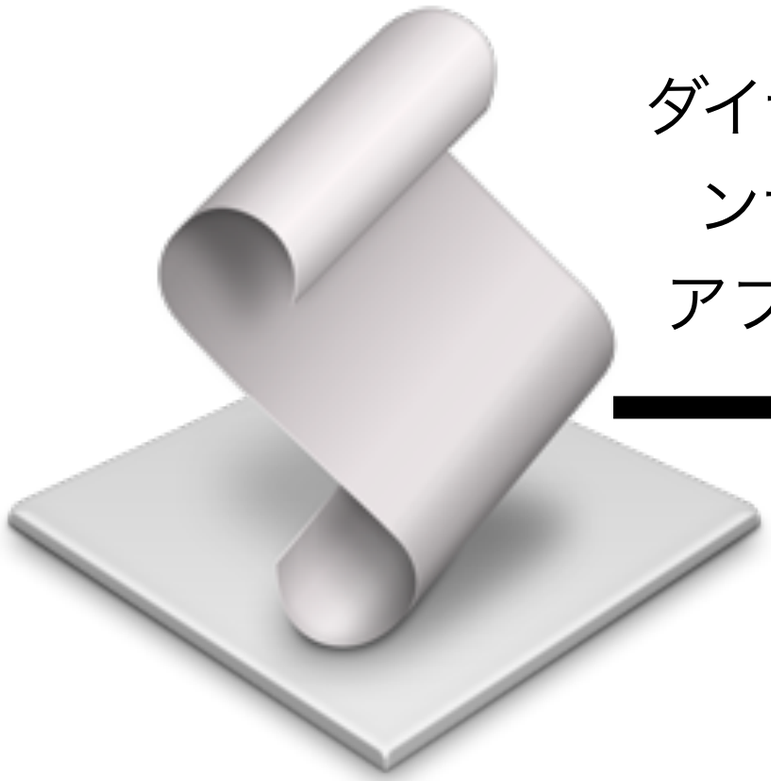
→　AppleScriptで全力で処理していても、多くのCPUコアが遊んでいる

→　もったいない！！！

狂気の産物、並列処理ごっこ（2011年）

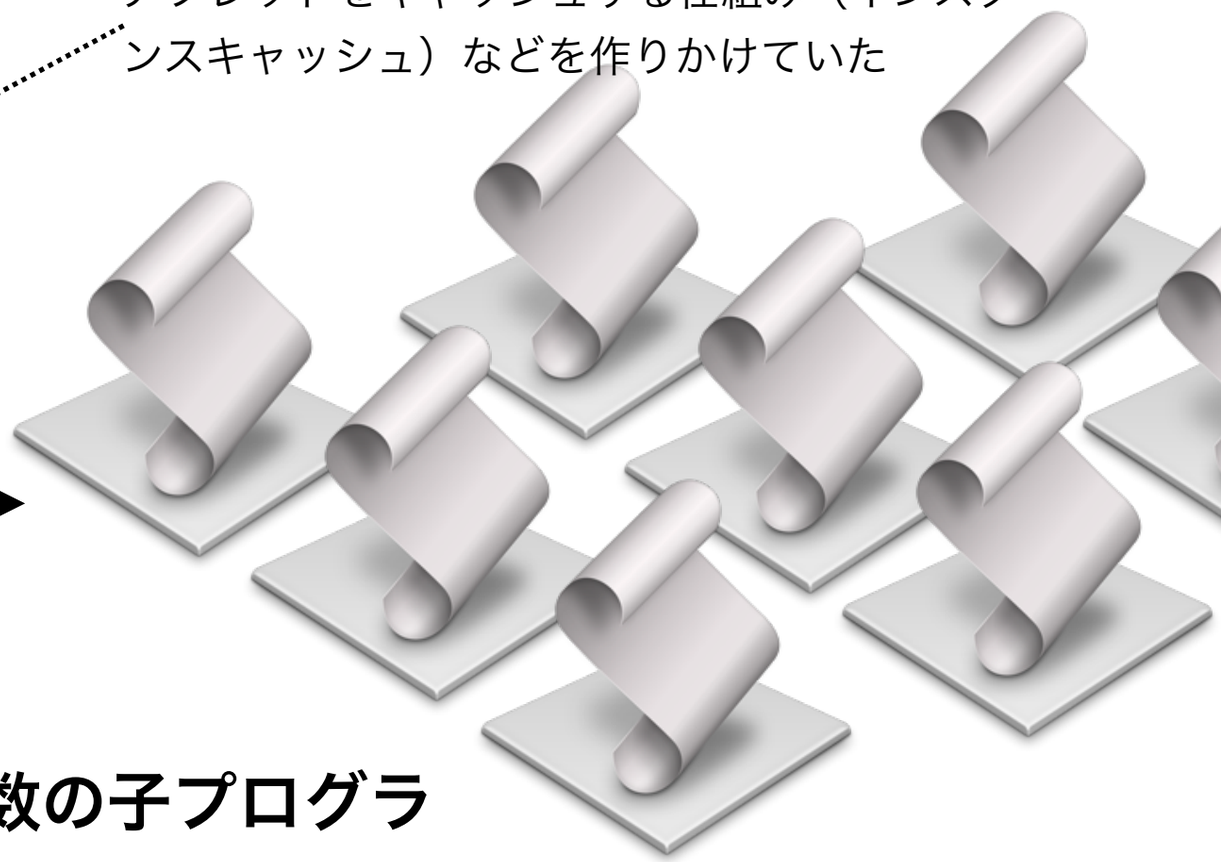
冗談半分で作った動的な並列処理メカニズム

アプレットをキャッシュする仕組み（インスタンスキャッシュ）などを作りかけていた



メインプログラム

ダイナミックに（オンザフライで）アプレットを生成



複数の子プログラムで並列実行

当時はまだ**HDD**上で処理していたので、ダイナミックに実行プログラムを生成する部分のオーバーヘッドが大きかった

大量のEPSファイルの破損チェックなどの特定作業に威力を発揮

真剣にCocoa的なアプローチで並列処理対応してみた

2011年当時は使えなかった「Cocoa呼び出し」がOS X 10.10からできるようになったので

NSOperationQueueを使って スレッド処理

parallel_image_resizing — 編集済み

AppleScript | on _aHandler

```
-- Created 2015-08-22 by Takaaki Naganoya
-- 2015 Piyomaru Software
use AppleScript version "2.4"
use scripting additions
use framework "Foundation"
use framework "AppKit"
use framework "QuartzCore"


set aFol to (choose folder)
tell application "Finder"
	tell folder aFol
		set aList to every file as alias list
	end tell
end tell

set a1Dat to current application's NSDate's timeIntervalSinceReferenceDate()

--Create Queue
set aQueue to current application's NSOperationQueue's new()

set idCount to 1
repeat with i in aList

	set aPath to POSIX path of i
	set aDict to (current application's NSDictionary's dictionaryWithDictionary:{anID:idCount, aPath:aPath})
	set anOperation to (current application's NSInvocationOperation's alloc()'s initWithTarget:me
		selector:"_aHandler:" object:aDict)

	(aQueue's addOperation:anOperation)

	set idCount to idCount + 1
```

OSAScriptを使ってメモリ上で 複数インスタンス生成＋実行

parallel_OSAScript

AppleScript | <要素が選択されていません>

```
-- Created 2015-08-22 by Shane Stanley

use AppleScript version "2.4"
use scripting additions
use framework "Foundation"
use framework "OSAKit"

-- set the above to run after 1 second
set aStr to current application's NSString's stringWithString:"2nd"
my performSelector:"doItInTheBackground:" withObject:(aStr) afterDelay:1
set aStr to current application's NSString's stringWithString:"3rd"
my performSelector:"doItInTheBackground:" withObject:(aStr) afterDelay:1

-- start a repeat loop
repeat with i from 1 to 20
	delay 0.1
	current application's NSLog("On default instance, first thread %d", i) -- writes to Console
end repeat


on doStuff:aObject

	set aText to aObject as text
	-- create a new AppleScript instance
	set anOSALanguageInstance to current application's OSALanguageInstance's
		languageInstanceWithLanguage:(current application's OSALanguage's defaultLanguage())

	-- make script and run it from the new instance
	set theSource to "use framework \"Foundation\"
repeat with i from 1 to 20
 delay 0.1
 current application's NSLog(\"On " & aText & " instance, second thread %d\", i) -- writes to Console
end repeat"
```

どちらも、**普通に逐次処理したほうが高速**というレベル

（並列処理したら、逆に遅くなった）

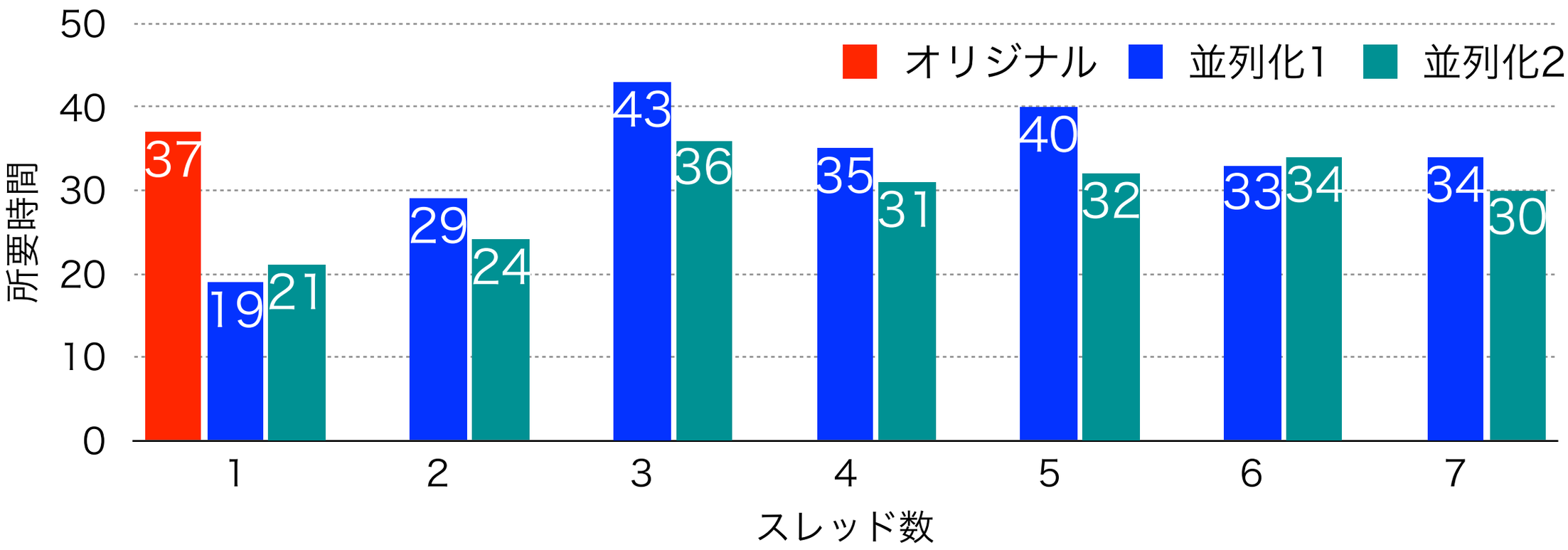
誰も予想しなかった「まさか」の展開

真剣にCocoa的なアプローチで並列処理

「並列処理ごっこ」を高度化して並列処理

並列処理化の効果

CoreImageを用いて75個の画像ファイルにフィルター①＋リサイズ＋フィルター②



そんなに速くなっていない（おいおい！）

なぜかスレッド数＝１のときが最速（Turbo Boostの効果？）

Disk I/Oがボトルネックになっている可能性は否定できない（SSDだけど）

MacBook Proで実施したので、サーマルスロットリング発生の可能性も否定できない。デスクトップ機のCPUで再試行したい（もってないけど）