B-001

モバイル端末の OS とブラウザの組合せに対するベンチマーク結果に関する考察
A Benchmark Analysis for Combination with Mobile Devices, OS, and Browsers

田島誠也 † テーム レッペネン § 岩井将行 ‡
Seiya Tajima Teemu Leppanen Masayuki IWAI

1. はじめに

モバイル情報端末におけるアプリケーション（以下，アプリとする）は一般に広く普及し，利用されている．その多くは端末上で動いている OS ごとにマーケットを通してインストールする形式を取っており，個々の端末へのインストールを前提としている．対して，近年の HTML や CSS，JavaScript のリファレンスや API の整備により，アプリ環境として Web ブラウザ及びコンポーネントを用いた Web アプリも，Microsoft Office Web Apps や Google mail など，PC を主な対象として普及してきている．Web アプリは，ハードウェアや OS などの基幹アーキテクチャによってソフトウェア実装の言語を変える必要がなく，モダンなブラウザが動作するすべてのプラットフォーム上で実行可能な特性から，注目を集めている．

2.JavaScript の動作環境

JavaScript は，Web アプリにおけるユーザエクスペリエンスの向上において動的性質の付与のために重要である．一方で，弱い動的型付けの言語であり，実行時に逐次解釈する仕様から，アプリ実行時のボトルネックとなっている．この解消のため，ブラウザごとに様々な処理系が実装されているほか，高速化のための研究 [1][2] が行われている．本稿ではモバイル情報端末上で JavaScript を用いたベンチマークテストを行い，その結果からモバイル情報端末における Web アプリの動作環境について考察する．

3. モバイル環境における JavaScript 評価手法

実機端末上でブラウザを立ち上げ，端末・OS・ブラウザを 1 つの組合せとする．組合せごとに Octane[3] にてベンチマークテストを行い，そのスコアを記録する．組合せごとのスコアを比較することで，評価を行う．評価対象とする組合せとその番号を表 1 に示す．評価対象端末の CPU，RAM を表 2 に示す．

3.1.Octane JavaScript Benchmark

今回は，ブラウザを用いたベンチマークテストを行うために Octane を使用した．Octane は，Google 製の JavaScript ベンチマークスイートであり，それまでのベンチマークテストよりもより現実的なアプリの動作をエミュレートすることを目的として開発された．13 のベンチマークテストモジュールに分かれており，ベンチマークが終了するとそれぞれのスコアと総合スコアが表示される．

†東京電機大学工学部第二部情報通信工学科，TDU
‡東京電機大学未来科学部情報メディア学科，TDU
§The University of Oulu, The University of Tokyo

表 1: 評価対象とする組合せ

No.	端末名	OS Version	Browser
1	Nexus S	Android 4.0.4	Chrome 27.0.1453
2	Nexus S	Android 4.0.4	Firefox 22
3	Nexus S	Firefox OS 1.1.0.0-pre	default
4	iPhone 5	iOS 6.1.3	Safari
5	iPhone 5	iOS 6.1.3	Chrome 27.0.1453
6	Galaxy S4	Android 4.2.2	default
7	Galaxy S4	Android 4.2.2	Chrome 27.0.1453

表 2: 評価対象端末の CPU，RAM

端末名	CPU	RAM
Nexus S	Single-core 1 GHz	512MB
iPhone 5	Dual-core 1.2 GHz	1GB
Galaxy S4	Quad-core 1.9GHz	2GB

3.2. 評価システムの構築

ベンチマークテストの結果は，ブラウザで Octane[3] にアクセスし,「Start Octane」を押すことでベンチマークテストが行われた後に結果が表示される．しかし，1 度のベンチマークテストでは端末の状態等による値の上下があり，比較時の指標に用いるには複数回の値を取る必要があった．今回は，連続的にベンチマークテストを行い結果を記録するために，簡易な HTTP サーバを Ruby でプログラムした．この HTTP サーバにアクセスする URL によりベンチマーク回数を決定する．ブラウザは，ベンチマークテストが 1 回終了するごとに，Ajax 技術を用いて各ベンチマークスコアの情報をサーバに POST する．サーバ側では受け取った値を CSV に書き出すようにした（図 1）．

表 3: Benchmark モジュールの簡易説明

Name	Benchmark Type
Richards	OS kernel simulation
Deltablue	One-way constraint solver
Raytrace	Ray tracer
Regexp	Regular expression
NavierStokes	2D NavierStokes equations solver
Crypto	Encryption and decryption
Splay	Data manipulation
EarleyBoyer	Classic Scheme
pdf.js	PDF Reader
Mandreel	3D Bullet Physics Engine.
GB Emulator	portable console's architecture
Code loading	quickly engine start after loading
Box2DWeb	popular 2D physics engine

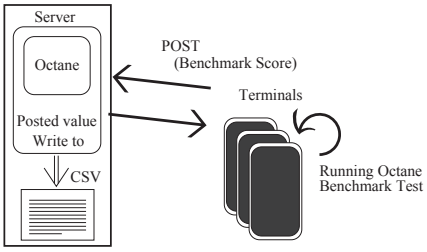

図 1: 構築した評価システムの概要図

3.3. モバイルオペレーティングシステムの書き換え

Firefox OS は，Mozilla が中心となりモバイル端末向けに開発した OS であり，Web 技術のみでアプリが動作することを特徴としている．プラットフォームとして Firefox ブラウザの技術を採用している．Firefox OS については，2013 年 6 月現在国内キャリアから発売されておらず，今回評価を行うに際して自ら用意する必要があった．build し，それを Nexus S 端末にフラッシュすることで Firefox OS 端末の代用とした．

4. 評価結果

評価として，1 つの組合せにつき 50 回ずつ評価を行い，それらの平均を指標として比較する（図 2）．しかし，No.2 においては，ベンチマークテスト実行中にブラウザがクラッシュしてしまい，テスト結果を記録することが出来なかった．

ハードウェアから見る差異

No.1 対 No.6 の比較から，ハードウェア性能の違いによる差異を見ることが出来る．おおよそ CPU 動作速度やコア数に比例してスコアがあがっていることがわかる．今後も CPU コア数の増加や省電力化により携帯情報端末の処理能力は向上していくものと思われる．

OS から見る差異

No.1 及び No.6 対 No.5，No.1 及び No.2 対 No.3 の比較から，OS 仕様の違いによる差異を見ることが出来る．Google Chrome は，PC 及び Android においては，JavaScript 処理系として v8 JavaScript Engine を開発・採用している．しかし，現在の iOS における制約として，Safari は Nitro と呼ばれる JavaScript 処理系を使用しているが，Apple 以外のサードパーティ開発者の使用できる JavaScript 処理系は UIWebView Class API のみである．このため，このように大きな差が現れたものと思われる．現状において，iOS を対象に含む Web アプリを開発する際は，JavaScript 処理の量にもよるが，UserAgent における判別から，処理内容を簡易なものに切り替える，もしくは Safari ブラウザにて再度アクセスするよう促す，等の必要がある．また，同一ハードウェアを使用しても，ブラウザがクラッシュする差異を確認した．No.2 において，ブラウザがクラッシュするタイミングがある程度同一のタイミングであったため，一回にベンチマークを行うモジュールを段階的に減らしたところ，ある程度まで減らすことでブラウザのクラッシュを避けることが出来た．これは，Android と Firefox OS のアーキテクチャの違いによるものだと思われる．ただし，今回使用した端末は製品として提供されたものではないため，最適化が行われていないことに留意しなければならない．

5. おわりに

本稿では，Octane を用いて Web アプリの実行環境について考察した．モバイル情報端末は年々高性能になってゆくが，新しい端末・技術によりユーザ環境の細分化が進んでしまう．ハードウェアリソースの少ないユーザ環境を考慮すると，HTML，CSS，JavaScript を適切に使い，ユーザ環境における演算処理量を減らす必要がある．また，単一の Web アプリにおいて，ユーザエクスペリエンスを提供する際には UserAgent やブラウザの機能検知により，処理内容を変更する必要がある．尚，本研究は H25 科研費若手研究 (A)(代表者: 岩井将行, 課題番号:25700007) の一部により行われている．

参考文献

[1] 田村知博，中野圭介，鵜川始陽，岩崎英哉，「JavaScript におけるプログラム変換の効果」，夏のプログラミング・シンポジウム 2011 報告集 pp19-26, 2011

[2] Francesco Logozzo, Herman Venter, 「RATA: Rapid Atomic Type Analysis by Abstract Interpretation. Application to JavaScript optimization.」, Proceedings of the International Conference on Compiler Construction pp66-75, 2010

[3] Octane JavaScript Benchmark, `http://octane-benchmark.googlecode.com/svn/latest/index.html`

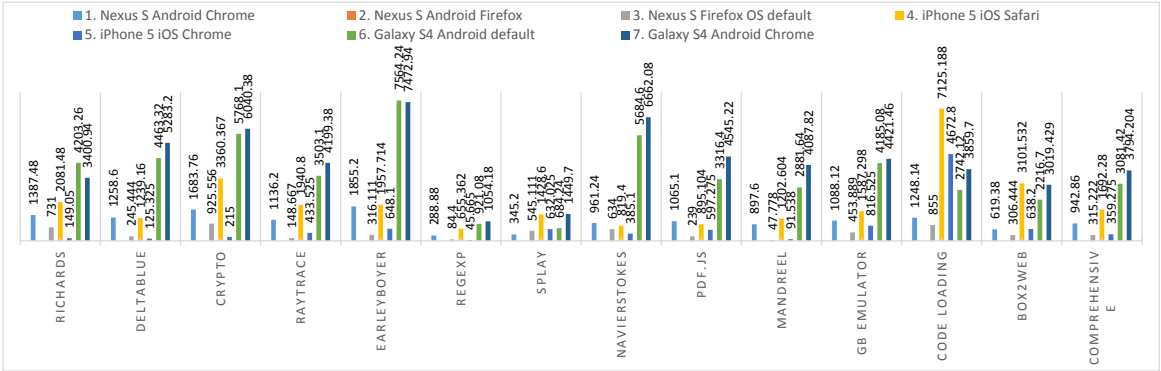

図 2: Average of Octane Benchmark's Score