Vesper: Measuring Time-to-Interactivity for Web Pages

Ravi Netravali*  Vikram Nathan*  James Mickens‡  Hari Balakrishnan*

*MIT CSAIL  ‡Harvard University
The Importance of Page Load Time

Slow page loads → lost revenue and low search rank

Research: Site Speed Is Hurting Your Everyone’s Revenue
IAN LURIE // MAY 9 2014

Site speed, site speed, site speed. Everyone around me is sick of hearing me rant about it because I’ve pushed it on every client Portent’s had since, oh, 2008.

Google Play will now downrank poorly performing apps
Posted Aug 3, 2017 by Sarah Perez (@sarahintampa)

Google has kept a promise: It made last year. Site speed is now a ranking factor in Google’s algorithm, and is already in place for U.S. searchers. But Google also cautions web site owners not to sacrifice relevance in the name of faster web pages, and even says this new ranking factor will impact very few queries. More on that below, but first the background on today’s announcement from Google Fellow Amit Singhal and Matt Cutts, head of Google’s web spam team.

Why Page Speed Matters
The first warning that site speed was on Google’s radar came last November, when Cutts said there

How One Second Could Cost Amazon $1.6 Billion In Sales

It’s Official: Google Now Counts Site Speed As A Ranking Factor
BY TONY PALAZZO
Business.com / Marketing Solutions / Last Modified: March 28, 2018

Google uses a multitude of factors to determine how to rank search engine results. These factors are either related to the content of a webpage itself (the text, its URL, etc.) or user-generated feedback (like social media links). In the case of Google, page speed is one of the factors that impacts search results. This is why it’s important to ensure that your website loads quickly.

Here’s Everything You Need to Know About Google's ‘Speed Update’
Website loading speeds on mobile devices will now be used as ranking factors in Google search results. Earlier this year, Google announced a big change to the way they factor search rankings. For the first time, website loading speeds on mobile devices will be used as ranking factors in search results.

How Website Speed Actually Impacts Search Ranking
On-page SEO
The author’s views are entirely his or her own (excluding the unlikely event of hypnosis) and may not always reflect the views of Avo.
Everyone agrees that web pages should load quickly...
Everyone agrees that web pages should load quickly...

...but how should page load time be defined?
Contributions

1. **Ready Index (RI)**: analytical definition of page time-to-interactivity in terms of visibility and functionality

2. **Vesper**: system that automatically measures RI

3. **Optimizing pages for RI**: framework to optimize page loads for time-to-interactivity

4. **User studies**: interactive users strongly prefer pages that optimize for RI
Outline

• How pages load today
• Existing Metrics
• Ready Index (RI)
  – Definition
  – Measurement system (Vesper)
• Evaluation
  – RI vs. preexisting metrics
  – Optimizing pages for RI (time-to-interactivity)
  – User studies: how does RI capture user experience?
Page Loads Today
Page Loads Today
Page Loads Today

http://www.amazon.com
Page Loads Today

http://www.amazon.com

browser

amazon.com
Page Loads Today

http://www.amazon.com

browser

amazon.com
Page Loads Today

http://www.amazon.com

JavaScript Engine

```
var x = 10;
var y = {'prop1': x};
```

JavaScript Heap

amazon.com

browser
Page Loads Today

http://www.amazon.com

JavaScript Engine

```javascript
var x = 10;
var y = {'prop1': x};
```

JavaScript Heap

Rendering Engine

DOM tree (representation of HTML)
Page Loads Today

http://www.amazon.com

JavaScript Engine

var x = 10;
var y = {'prop1': x};

JavaScript Heap

DOM Interface

Rendering Engine

<html>
<head>
<body>

DOM tree (representation of HTML)

browser

amazon.com
Page Loads Today

http://www.amazon.com

JavaScript Engine
var x = 10;
var y = {'prop1': x};

JavaScript Heap

DOM Interface

Rendering Engine

DOM tree (representation of HTML)

browser

amazon.com

static.amazon.com
Page Loads Today

http://www.amazon.com

amazon.com

static.amazon.com

browser
Existing Metrics
Existing Metrics

DOM Tree

JavaScript heap
Existing Metrics

Page load time (PLT): time until all objects are fetched and evaluated

DOM Tree

JavaScript heap
Existing Metrics

Page load time (PLT): time until all objects are fetched and evaluated
Existing Metrics

Page load time (PLT): time until all objects are fetched and evaluated

- Too conservative
Existing Metrics

Page load time (PLT): time until all objects are fetched and evaluated
- Too conservative

Speed Index (SI): time to render above-the-fold
**Existing Metrics**

**Page load time (PLT):** time until all objects are fetched and evaluated

- Too conservative

**Speed Index (SI):** time to render above-the-fold
**Existing Metrics**

**Page load time (PLT):** time until all objects are fetched and evaluated

- Too conservative

**Speed Index (SI):** time to render above-the-fold
Existing Metrics

Page load time (PLT): time until all objects are fetched and evaluated
  – Too conservative

Speed Index (SI): time to render above-the-fold
Existing Metrics

**Page load time (PLT)**: time until all objects are fetched and evaluated
- Too conservative

**Speed Index (SI)**: time to render above-the-fold
- Ignores JavaScript that supports functionality
**Interactive page:** above-the-fold is visually ready and fully functional
Interactive page: above-the-fold is visually ready and fully functional.
Interactive page: above-the-fold is visually ready and fully functional

Challenge: nobody knows a good way to automatically identify that interactive state
**Interactive page:** above-the-fold is visually ready and fully functional

**Challenge:** nobody knows a good way to automatically identify that interactive state

1. Identify page’s interactive state in DOM/JS
2. Analytically define rate at which state is visible and functional

![Graph showing cumulative distribution function (CDF) with median and 95th percentile values.]

- Median: 182 handlers
- 95th percentile: 1252 handlers
Outline

• How pages load today
• Existing Metrics
• Ready Index (RI) + Vesper
• Evaluation
Ready Index
Functionality: $F(e, t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$

time when e’s JavaScript handlers are registered, and state that handlers access when fired is loaded
Functionality: \( F(e, t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases} \)

Visibility: \( V(e, t) = \frac{|P_t(e)|}{|P(e)|} \)

- \( e \) = above-the-fold element
- \( t_e \) = time when \( e \)'s handlers are registered, and state they access when fired is loaded

\( e \)'s paint events that are finished by time \( t \)

paint events that affect \( e \)
**Functionality:** \( F(e, t) = \begin{cases} 
0 & t < t_e \\
1 & t \geq t_e 
\end{cases} \)

- \( e \) = above-the-fold element
- \( t_e \) = time when \( e \)'s handlers are registered, and state they access when fired is loaded

**Visibility:** \( V(e, t) = \frac{|P_t(e)|}{|P(e)|} \)

- \( P(e) \) = paint events that affect \( e \)
- \( P_t(e) \) = \( e \)'s paint events that are finished by time \( t \)

**Element Readiness:** \( R(e, t) = \frac{1}{2} F(e, t) + \frac{1}{2} V(e, t) \)
Ready Index

Functionality: \( F(e, t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases} \)
- \( e \) = above-the-fold element
- \( t_e \) = time when \( e \)'s handlers are registered, and state they access when fired is loaded

Visibility: \( V(e, t) = \frac{|P_t(e)|}{|P(e)|} \)
- \( P(e) \) = paint events that affect \( e \)
- \( P_t(e) \) = \( e \)'s paint events that are finished by time \( t \)

Element Readiness: \( R(e, t) = \frac{1}{2} F(e, t) + \frac{1}{2} V(e, t) \)

Page Readiness: \( R(t) = \sum_{e \in E} A(e) R(e, t) \)
- Pixel area of \( e \)
Functionality: $F(e, t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$
- $e =$ above-the-fold element
- $t_e =$ time when $e$’s handlers are registered, and state they access when fired is loaded

Visibility: $V(e, t) = \frac{|P_t(e)|}{|P(e)|}$
- $P(e) =$ paint events that affect $e$
- $P_t(e) =$ $e$’s paint events that are finished by time $t$

Element Readiness: $R(e, t) = \frac{1}{2} F(e, t) + \frac{1}{2} V(e, t)$

Page Readiness: $R(t) = \sum_{e \in E} A(e) R(e, t)$
- $A(e) =$ pixel area of $e$

Ready Index: $R^I = \int_0^T 1 - \frac{R(t)}{R(T)} d(t)$
- loose upper bound on load time
Ready Index

**Functionality:** $F(e, t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$
- $e =$ above-the-fold element
- $t_e =$ time when e’s handlers are registered, and state they access when fired is loaded

**Visibility:** $V(e, t) = \frac{|P_t(e)|}{|P(e)|}$
- $P(e) =$ paint events that affect $e$
- $P_t(e) =$ e’s paint events that are finished by time $t$

**Element Readiness:** $R(e, t) = \frac{1}{2} F(e, t) + \frac{1}{2} V(e, t)$

**Page Readiness:** $R(t) = \sum_{e \in E} A(e) R(e, t)$
- $A(e) =$ pixel area of $e$

**Ready Index:** $RI = \int_0^T 1 - \frac{R(t)}{R(T)} d(t)$
- $T =$ loose upper bound on load time

**Ready Time (RT):** smallest time when all above-the-fold elements are ready
Measuring Ready Index (RI)
Measuring Ready Index (RI)

• Need to know:
  Visible elements and their event handlers
  State that handlers access when fired
  Effect and timing of browser paint events
Measuring Ready Index (RI)

• Need to know:
  Visible elements and their event handlers
  State that handlers access when fired
  Effect and timing of browser paint events

• Requirements for instrumentation:
  No developer annotations
  Low overhead
  Generic
Vesper: Overview

**Approach:** Use two measurement phases to reduce impact of instrumentation

**Phase 1 (offline):** Identify page’s interactive state

**Phase 2:** Track loading progress of interactive state

List of interactive DOM nodes and JavaScript state
Vesper: Phase 1

**Goal:** Identify visible elements, event handlers, and the state handlers access when fired
Vesper: Phase 1

**Goal**: Identify visible elements, event handlers, and the state handlers access when fired
Vesper: Phase 1

**Goal:** Identify visible elements, event handlers, and the state handlers access when fired

*Above The Fold*

**Element visibility:** analyze element bounding boxes and CSS rules

*DOM Tree*

*JavaScript heap*
Vesper: Phase 1

**Goal**: Identify visible elements, event handlers, and the state handlers access when fired

- **Element visibility**: analyze element bounding boxes and CSS rules

- **Logging event handlers**: shim event handler registration mechanisms
Vesper: Phase 1

**Goal:** Identify visible elements, event handlers, and the state handlers access when fired

- **Element visibility:** analyze element bounding boxes and CSS rules
- **Logging event handlers:** shim event handler registration mechanisms
- **Event handler state:** fire handlers and log accessed state with Scout

*DOM Tree*

*Above The Fold*
**Goal**: Identify visible elements, event handlers, and the state handlers access when fired

**Element visibility**: analyze element bounding boxes and CSS rules

**Logging event handlers**: shim event handler registration mechanisms

**Event handler state**: fire handlers and log accessed state with Scout

**Phase 1**: 4.5% overhead
Vesper: Phase 2

**Goal**: Track loading progress of interactive state from Phase 1
Vesper: Phase 2

**Goal:** Track loading progress of interactive state from Phase 1

DOM Tree

JavaScript heap
Vesper: Phase 2

**Goal:** Track loading progress of interactive state from Phase 1
**Goal**: Track loading progress of interactive state from Phase 1
Vesper: Phase 2

**Goal:** Track loading progress of interactive state from Phase 1
Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1

• Log “last writes” for DOM/heap state
Goal: Track loading progress of interactive state from Phase 1

- Log “last writes” for DOM/heap state

```javascript
var x = 50; var y = 0;
while (y < 50) {
  x = x + 1;
  y = y + 1;
}
x = x + 5;
```
Vesper: Phase 2

**Goal:** Track loading progress of interactive state from Phase 1
- Log “last writes” for DOM/heap state

```javascript
var x = 50; var y = 0;
while (y < 50) {
  x = x + 1;
  y = y + 1;
  if (y == 49) {
    vesper_log(y);
  }
}
x = x + 5;
vesper_log(x);
```
Goal: Track loading progress of interactive state from Phase 1

- Log “last writes” for DOM/heap state
- Track browser layout/paint events

```javascript
var x = 50; var y = 0;
while (y < 50) {
    x = x + 1;
    y = y + 1;
    if (y == 49) {
        vesper_log(y);
    }
}
x = x + 5;
vesper_log(x);
```
Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1

- Log “last writes” for DOM/heap state
- Track browser layout/paint events

Phase 2: 1.9% overhead

```javascript
var x = 50; var y = 0;
while (y < 50) {
  x = x + 1;
  y = y + 1;
  if (y == 49) {
    vesper_log(y);
  }
}
x = x + 5;
vesper_log(x);
```
Outline

• How pages load today
• Existing Metrics
• Ready Index (RI) + Vesper
• Evaluation
Evaluation Outline

• Are there differences between Ready Index and existing metrics?

• Can we optimize a page load for Ready Index?

• How well does Ready Index capture user experience?
AFT vs. RT vs. PLT

Time to AFT: 3.40s
Time to RT: 5.96s
Time to PLT: 8.68s

- Search category dropdown menu
- Search bar
- Sign-in dropdown menu
- Cart button
- Search button

amazon.com
Above-the-fold time (AFT) underestimates ‘interactive time’ by 2.56 seconds!
Above-the-fold time (AFT) underestimates ‘interactive time’ by 2.56 seconds!

Page load time (PLT) overestimates ‘interactive time’ by 2.72 seconds!
350 Popular Sites: AFT vs. RT vs. PLT

12 Mbits/s, 50 ms RTT

PLT: 3.4 s
RT: 2.5 s
AFT: 1.9 s
350 Popular Sites: AFT vs. RT vs. PLT

12 Mbits/s, 50 ms RTT

PLT: 3.4 s
RT: 2.5 s
AFT: 1.9 s

- PLT > RT > AFT (differences of 24.0%-64.3%, 0.3-3.6 seconds)
350 Popular Sites: AFT vs. RT vs. PLT

PLT: 3.4 s
RT: 2.5 s
AFT: 1.9 s

• PLT > RT > AFT (differences of 24.0%-64.3%, 0.3-3.6 seconds)
• Differences increase as RTTs increase
350 Popular Sites: SI vs. RI

12 Mbits/s, 50 ms RTT

RI: 1759 ms
SI: 1325 ms
Optimizing for Ready Index

• Vesper: identify objects of importance
• Polaris: optimize loading of important objects
  • Dependency-aware request scheduler that uses dynamic critical path analysis to reduce page load times
Optimizing for Ready Index

- **Vesper**: identify objects of importance
- **Polaris**: optimize loading of important objects
  - Dependency-aware request scheduler that uses dynamic critical path analysis to reduce page load times
Optimizing for Ready Index

- Vesper: identify objects of importance
- Polaris: optimize loading of important objects
  - Dependency-aware request scheduler that uses dynamic critical path analysis to reduce page load times

```
Polaris_PLT
```

```
Polaris_SI
```
Optimizing for Ready Index

- Vesper: identify objects of importance
- Polaris: optimize loading of important objects
  - Dependency-aware request scheduler that uses dynamic critical path analysis to reduce page load times

```
Polaris_PLT
  1
  /|
  1 1
  /|
  1 1
  /|
  1 1

Polaris_SI
  5
  /|
  1 1
  /|
  1 1
  /|
  5 1

Polaris_RI
  5
  /|
  5 1
  /|
  5 1
```
# Optimization Results

12 Mbits/s, 100 ms

<table>
<thead>
<tr>
<th>Weight</th>
<th>PLT</th>
<th>RI</th>
<th>SI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polaris-PLT</td>
<td>36%</td>
<td>8%</td>
<td>-7%</td>
</tr>
<tr>
<td>Polaris-RI</td>
<td>23%</td>
<td>29%</td>
<td>12%</td>
</tr>
<tr>
<td>Polaris-SI</td>
<td>10%</td>
<td>14%</td>
<td>18%</td>
</tr>
</tbody>
</table>
## Optimization Results

12 Mbits/s, 100 ms

<table>
<thead>
<tr>
<th>Weight</th>
<th>PLT</th>
<th>RI</th>
<th>SI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polaris-PLT</td>
<td>36%</td>
<td>8%</td>
<td>-7%</td>
</tr>
<tr>
<td>Polaris-RI</td>
<td>23%</td>
<td>29%</td>
<td>12%</td>
</tr>
<tr>
<td>Polaris-SI</td>
<td>10%</td>
<td>14%</td>
<td>18%</td>
</tr>
</tbody>
</table>

Targeted metrics improve the most!
User Study 1: Interactivity

• Perform interactive task with Polaris-PLT, Polaris-RI, Polaris-SI: which is fastest?
• 5 sites, 85 users
User Study 1: Interactivity

• Perform interactive task with Polaris-PLT, Polaris-RI, Polaris-SI: which is fastest?

• 5 sites, 85 users

<table>
<thead>
<tr>
<th>Scheduling policy</th>
<th>Preference percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polaris-RI</td>
<td>83%</td>
</tr>
<tr>
<td>Polaris-SI</td>
<td>4%</td>
</tr>
<tr>
<td>Polaris-PLT</td>
<td>7%</td>
</tr>
<tr>
<td>None</td>
<td>6%</td>
</tr>
</tbody>
</table>

Takeaway: interactive users strongly prefer pages optimized for RI
User Study 2: Rendering

- 15 sites, 73 users
User Study 2: Rendering

- 15 sites, 73 users

Takeaway: Polaris-SI is best for rendering, but Polaris-RI is comparable.
Conclusion

• Existing web performance metrics ignore page interactivity
  – Over or underestimate time-to-interactivity by 24%-64%

• Ready Index (RI): analytical definition of page time-to-interactivity

• Vesper: system to automatically measure RI by identifying and tracking loading of page’s interactive state
  – Helps reduce time-to-interactivity by 32%