Doloto:

Code Splitting for AJAX Applications

Ben Livshits and Emre Kiciman
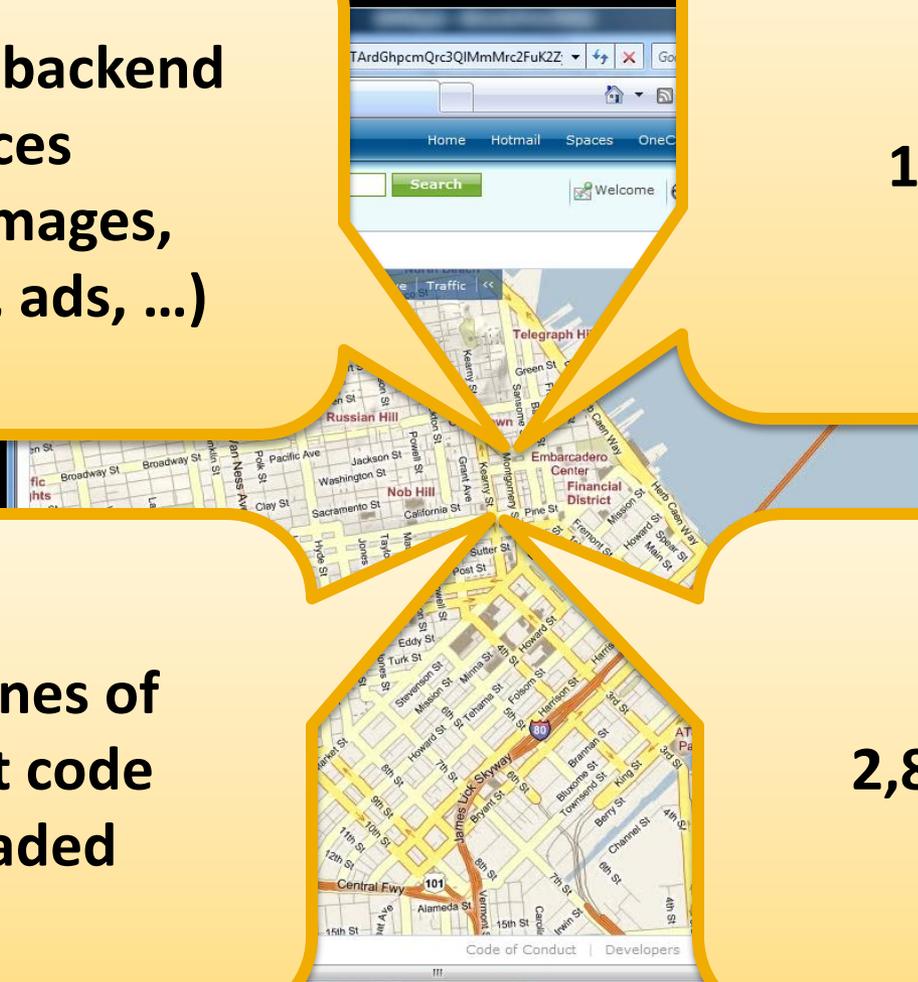**Microsoft Research**
Redmond, WA

A Web 2.0 Application Disected

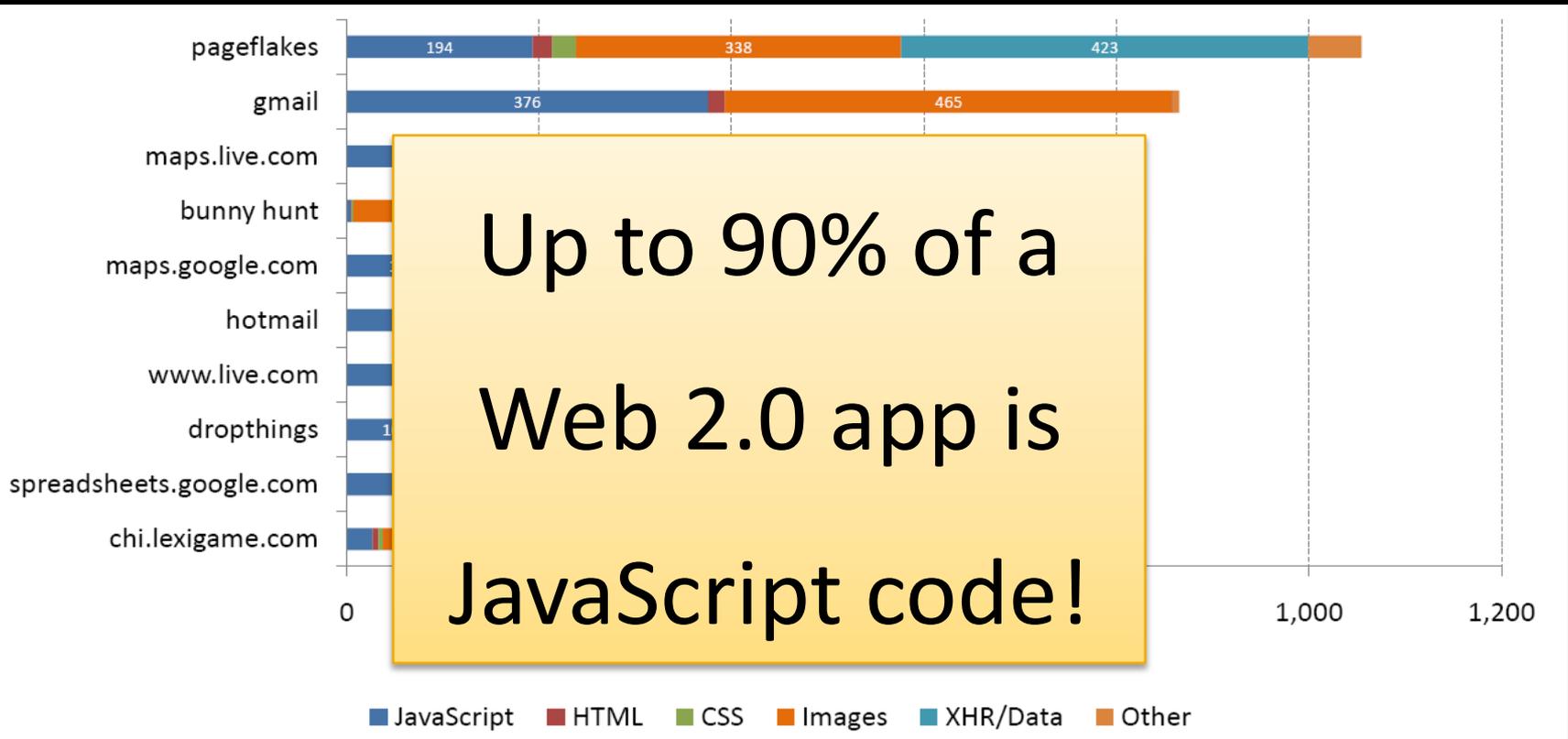**Talks to 14 backend services (traffic, images, directions, ads, …)**

1+ MB code

70,000+ lines of JavaScript code downloaded

2,855 Functions

AJAX Apps Have Lots of Code

Up to 90% of a

Web 2.0 app is

JavaScript code!

Why AJAX?

- Client-side JavaScript code
  - Reduces latency and improves user experience
  - Avoids constant network round trips

- Catch-22
  - Takes a while to transfer code to the client
  - Code can be MBs in size
  - Application execution is blocked until code arrives

Motivation for Doloto

- Idea behind Doloto
  - Start with a small piece of code on the client
  - Download required code on demand (pull)
  - Send code when bandwidth available (push)

- Leads to better application responsiveness
  - Interleave code download & execution
  - Faster startup times
  - Rarely executed code is rarely downloaded

Doloto Training Tool

Automated Function Splitting

```
var g = 10;
function f1(){
    var x=g+1;
    …
    …
    …
    …
    …
    return …;
}
```

```
var g = 10;

var real_f1;
function f1() {
    if(!real_f1){
        var code = load("f1");
        real_f1 = eval(code);
        f1 = real_f1;
    }
    return real_f1.apply(this,
                         arguments);
}
```
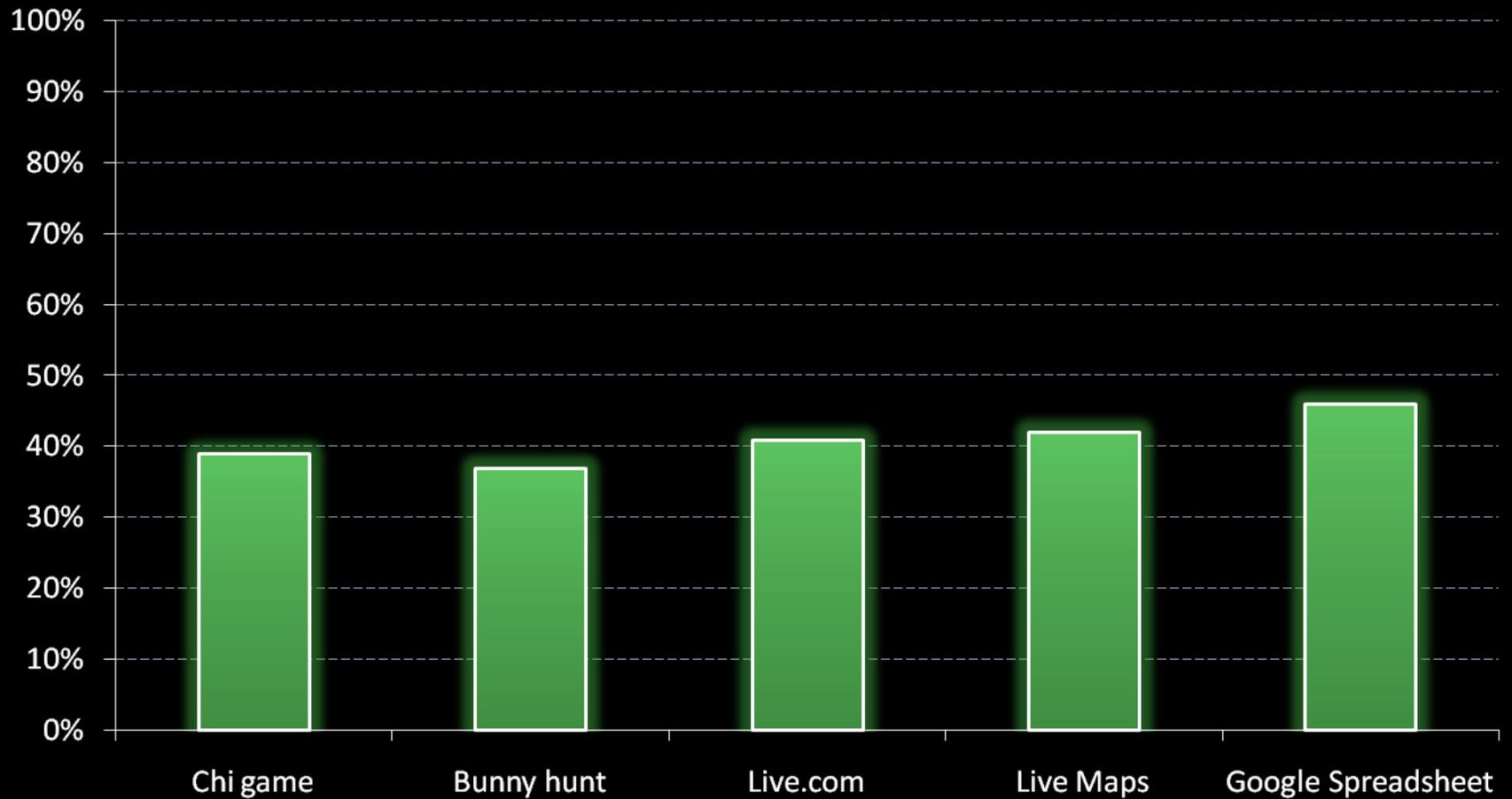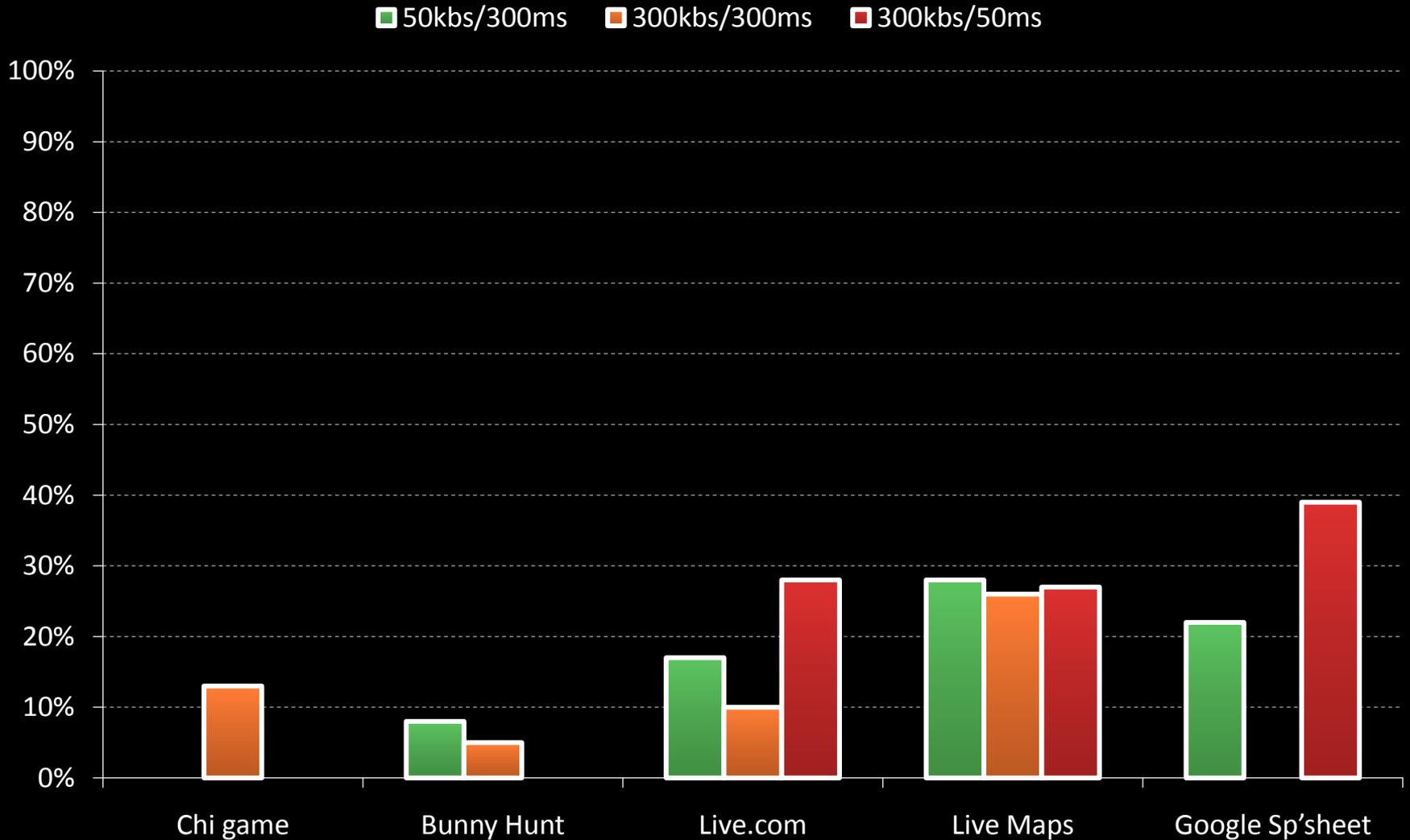
```
eval($exp("f1"), "");     // 21 chars
```

Architecture of Doloto

1. [**training**] Runtime training to collect access profiles

2. [**rewriting**] Function rewriting or "stubbing" for on-demand code loading

3. [**prefetch**] Background prefetch of clusters as the application is running

Size Savings with Doloto

Runtime Savings with Doloto

Contact us

Ben Livshits (livshits@microsoft.com)

Doloto MSR _