How to recoup the costs associated with long URLs and variable names

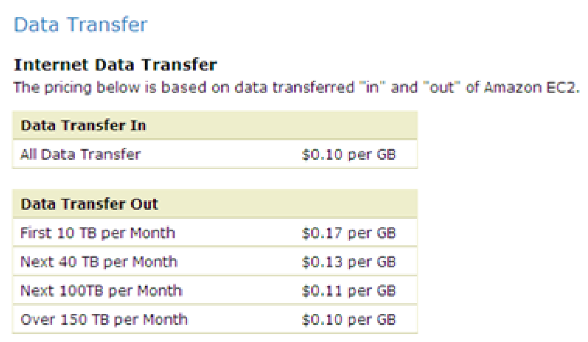
Lori MacVittie, 2009-31-03

Long URLs and variable names increase transfer size which wastes bandwidth and money

o3 magazine has a great article on the impact of long URLs on bandwidth; specifically on how much bandwidth is wasted by excessively long URLs and variable names within HTML, JavaScript, and CSS selectors.

What the author does not mention, and he really should, is that wasting bandwidth can translate into wasted dollars, as well. This is particularly true of applications that might be hosted in a cloud environment, as well as those delivered across WAN links provisioned with bursting capabilities above limits for which organizations are usually charged a premium price.

For example, using the analysis by o3 on the amount of "wasted" bandwidth and combing that with the cost per GB transferred through Amazon EC2 leads to a not insignificant dollar amount that is effectively thrown into the bit-bucket every month.



Data Transfer

Internet Data Transfer

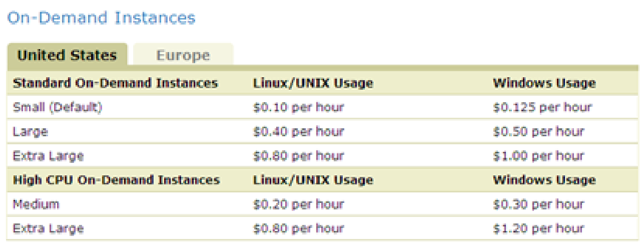
The pricing below is based on data transferred "in" and "out" of Amazon EC2.

Data Transfer In	
All Data Transfer	$0.10 per GB

Data Transfer Out	
First 10 TB per Month	$0.17 per GB
Next 40 TB per Month	$0.13 per GB
Next 100TB per Month	$0.11 per GB
Over 150 TB per Month	$0.10 per GB

Perhaps a valid point until the concept is applied to the 100+ character URL happy Facebook home.php page. There are roughly 150 source file references on this page, and rounding down to about 100 HREF requests for arguments sake. Being generous, assuming 80 bytes of waste per URL. Thats 12000 bytes of upstream, and 20000 bytes of downstream waste. So using data again from compete.com, facebook has 1,273,004,274 visits per month. This is roughly 41,064,654 requests per day. So on a single day, the folks over at facebook have wasted roughly 783GB downstream and 469GB upstream. This works out to be 74Mbit/sec downstream and 44MBit/sec upstream of bandwidth.

Based on the amount of "waste" and assuming Facebook was using EC2 instead of its own solution, this would translate into $158.03 a day for a total of approximately $4,740 / month. That would be nearly $56,780 a year wasted.



On-Demand Instances

United States | Europe

Standard On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Small (Default)	$0.10 per hour	$0.125 per hour
Large	$0.40 per hour	$0.50 per hour
Extra Large	$0.80 per hour	$1.00 per hour
High CPU On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**
Medium	$0.20 per hour	$0.30 per hour
Extra Large	$0.80 per hour	$1.20 per hour

But you're not Facebook, right? You'd never have that kind of traffic in a single day. Let's assume for a minute that those totals are *per month*, instead. In that case, you'd have wasted a mere $158.03 per month or approximately $2160 a year. Not quite so bad, right?

Before you dismiss that as irrelevant, let's translate *that* into the number of hours you could run your application on an EC2 instance.

Instance	**Hours**	**Days (24/7)**
Small Linux/UNIX	21600	900
Extra Large Linux/UNIX	2700	112.5
Small Windows	17280	720
Large Windows	2160	90

Essentially, if you're running a small Linux/UNIX instance then the money you could save from smaller URLs over one year would allow you to run that same instance for nearly 2.5 *years.*

Doesn't seem so irrelevant now, does it?

There are similar bandwidth-associated cost savings if you're leasing bandwidth and are charged a premium for bursting over your contracted bandwidth. The amount, of course, will vary based on what you're being charged for the overdraft on your bandwidth. A burstable T1 is common for small and medium sized businesses as it is highly cost effective, cutting the operating expense for a T1 about in half. But if you're constantly bursting at nearly full T1 capacity (1.544 Mbps) then you're going to end up paying more. Reducing the size of URLs and JavaScript variables can ensure that you stay in a lower pricing tier, thus ensuring that the recurring operating expense is reduced.

If you have access to a network-side scripting solution, you can automatically shorten URLs in application responses and subsequently map them back to the appropriate internal, long URL on request. The author of the article claims this puts undue stress on application delivery controllers, but rewriting URLs is one of the core capabilities for which an application delivery controller is optimized, so the burden isn't nearly as stressful as the author implies. Even assuming an addition of 2-3% utilization on the application delivery controller yields a higher benefit in saved bandwidth and operating expenses than any cost associated with such a slight increase. You can also use ***mod_rewrite*** to affect such a change if you're running Apache internally, which will also increase utilization of server resources but also provide similar benefits to an application delivery controller in terms of rewriting URLs.

The reasons for using a network-side scripting solution to make such changes to URLs generally revolve around the time and effort involved in rewriting the application. Such an effort may not be seen as having sufficient ROI on bandwidth savings to prioritize, so using a network-side scripting solution eliminates the need for developer resources (and associated testing and deployment costs) and affects change in a more immediate fashion.

If you're just starting development on a project, whether you're planning on hosting in the cloud or locally, consider the ramifications on bandwidth and associated costs from long URLs and excessively long variable names. Reducing the size of these aspects of an application up front can result in considerable savings in the long run.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com