

# **Geographic Distribution for Global Web Application Performance**



**Jacob Rosenberg  
Operations Architect**

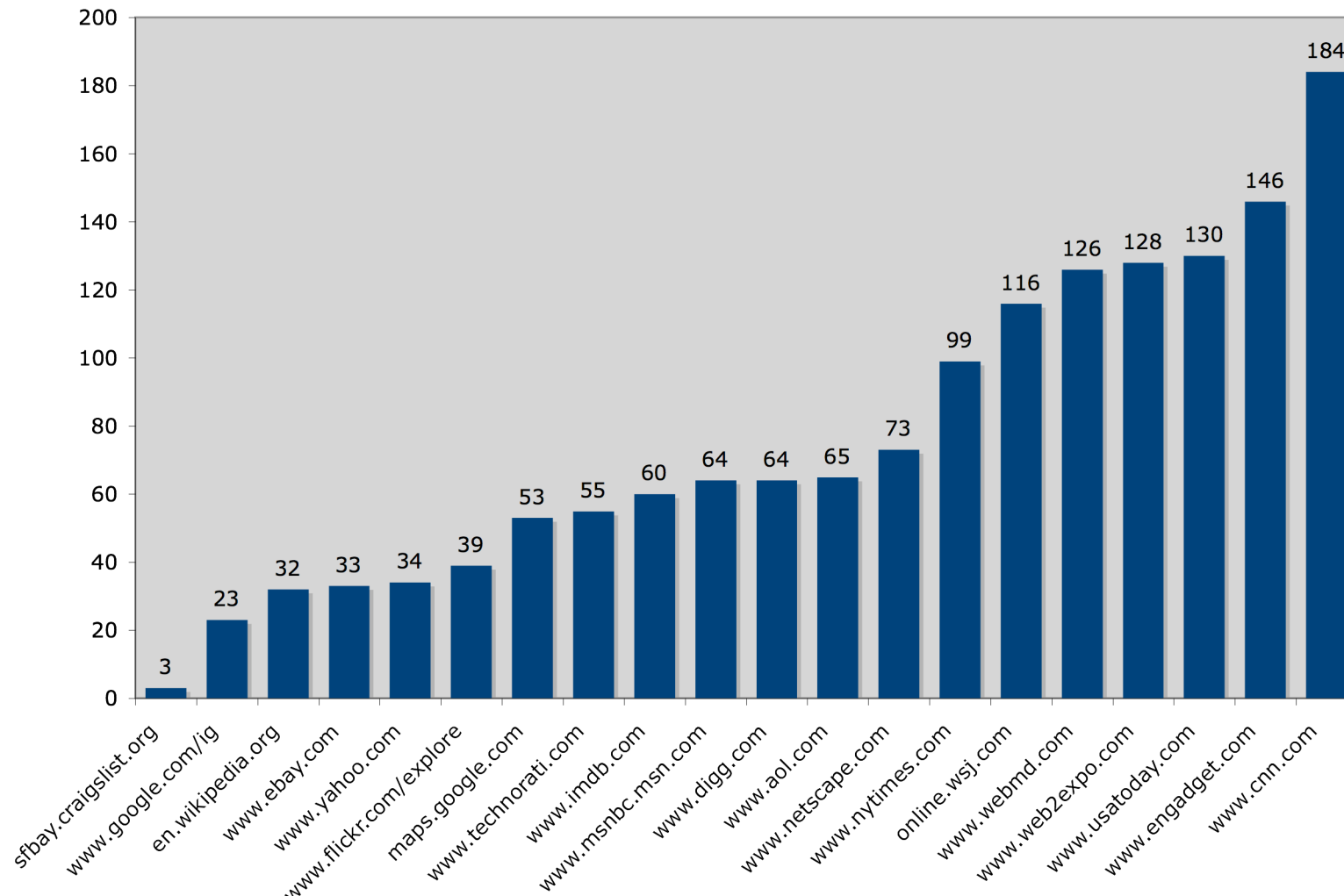
# Agenda

- **An Introduction to Geographic Distribution**
  - How modern web architecture makes latency important
  - Setting realistic goals for latency and application performance
- **Key Technique 1: Delivering static bits via content delivery networks**
  - Don't do more work than necessary - let content distribution networks deliver static bits of your site to your end users.
- **Key Technique 2: Going Multi-Site - Challenges and Opportunities**
  - Deliver your application simultaneously from multiple locations to improve the user experience and survive disasters.
- **Q&A**



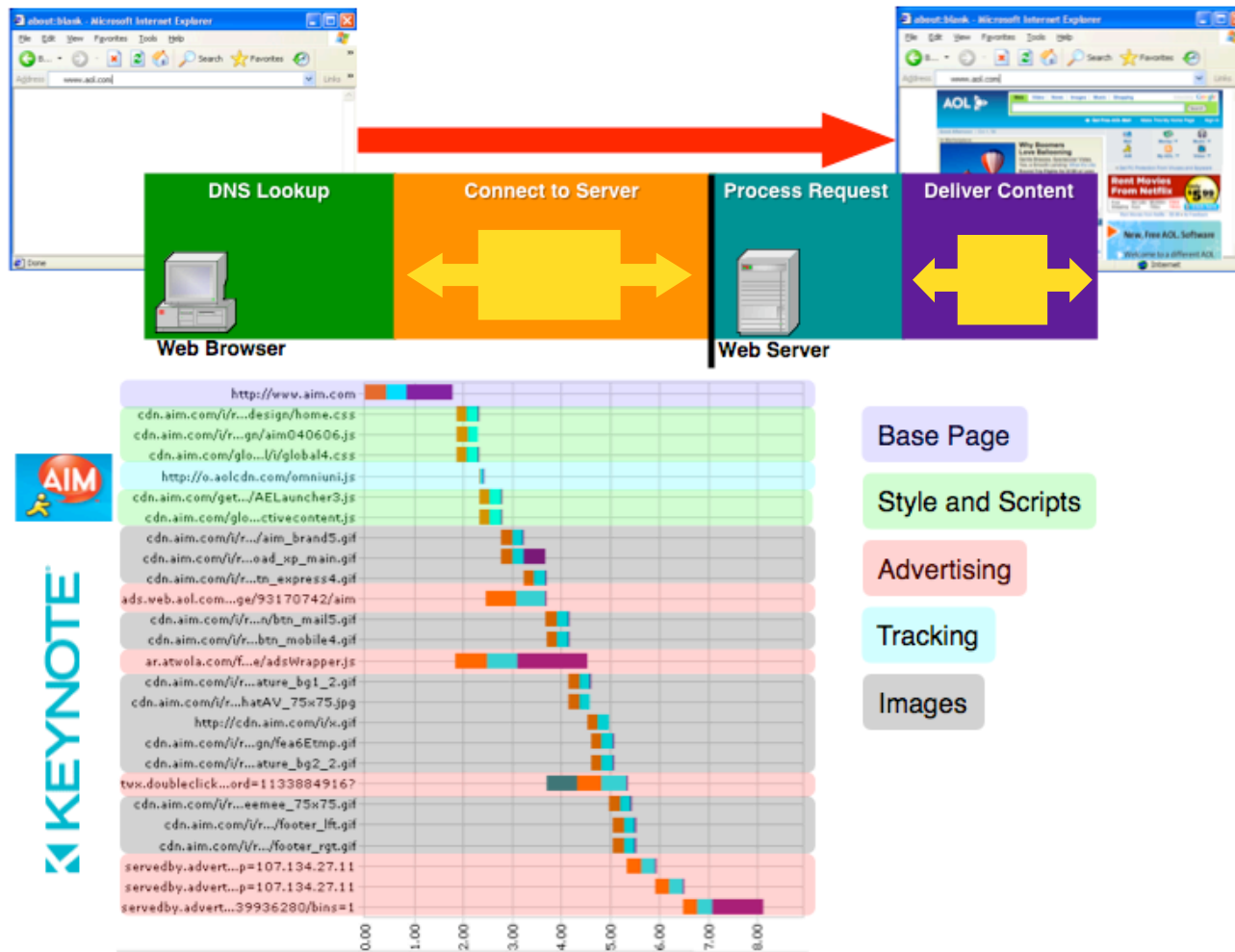
# Applications on the web are getting bigger

Most modern web pages are built out of dozens of individual objects...



# ► So, how does latency affect all these objects?

Initial connection and content delivery time are increased by latency



## ► So, what are the real numbers around latency?

**Example:** Keynote tests of a single-site web application (okay, it was Digg.com) delivered from the West Coast of the US to:

- **San Jose, CA: 403ms (.4 seconds)**
- **New York, NY: 1,993 ms (2.0 seconds)**
- **Frankfurt, Germany: 3,658 ms (3.7 seconds)**
- **Bangalore, India: 6,224 ms (6.2 second)**

### **Object Breakdown:**

- 66 total objects
- 54 were static
  - 1 CSS
  - 12 javascript
  - 41 images

# ▶ So, what should my latency goal be?

## - How fast is “good enough”?

- Akamai/Jupiter 4 second report: A study commissioned by Akamai found that users of commerce sites were unlikely to wait more than 4 seconds for a page. Full study can be found at <http://www.akamai.com/4seconds>
- Usability Researchers: Dr. Jakob Nielsen offers up basic advice for usability at <http://www.useit.com/papers/responsetime.html> suggesting 0.1 seconds for direct UI manipulation (e.g., drop or drag, fiddle with an AJAX widget, etc), and a 1 second limit for a sense that the user is “freely navigating the command space”.

## - Check it out yourself

- View your site with a latency emulator
- Recruit a geographically diverse beta community
- Remember: nobody every complains a site runs too quickly

## - Rough guidelines

- First impressions are important: get the site entry point under 2 seconds
- AJAX applications go from cool to sluggish around 100ms per request
- Plan to use big multimedia? Send down a 25k loader, and fetch the rest later.
- Make sure to test on low speed links -- modems may be history but crowded WiFi can be worse



## Techniques to Reach Web Performance Goals

## ▶ Technique 1: Deliver static bits via Content Delivery Networks

### • What is a Content Delivery Network?

- For the purpose of this discussion, a CDN is a system of **Web Cache Servers** offered as a service to allow parts of your web application to be delivered to your end-users from a **network location with better proximity**.
- Differing from a regular web cache (which fetches pages from any internet sites for end-users at a specific location or ISP), a CDN is a **reverse proxy** which caches content from a specific publisher to be delivered to users on the web at large.
- To improve performance on your site, a CDN will need to have **multiple distribution points in geographically diverse areas** near your intended user communities, and some technology to **localize** users to the closest site.
- Many companies operate CDNs commercially -- the largest is still Akamai. In addition, some sites build their own smaller scale CDN -- Wikipedia is a good example of this as they reveal all their hosting details.
- In addition to providing performance improvement for your users, the CDN will offload traffic from your server complex (called the origin servers) which may result in reduced hosting costs or increased capacity.

## ▶ Technique 1: Deliver static bits via Content Delivery Networks

- **What do I do to implement a CDN?**

- Establish a relationship with a CDN provider of your choice.
- Select an alternate name from which to serve static content.
  - If your site is at `www.mysite.com`, consider `cdn.mysite.com`
  - Preferably, chose an alternate base domain and keep it cookie free
- Provision the name on your CDN provider via their process. Typically, provide:
  - The new hostname you have selected
  - The hostname where the content lives now (called the origin server)
  - Let them know how long to cache your content
- When provisioning is complete, they will ask you to make DNS changes.
- Finally, modify your site so that content is sourced from the CDN:  
``  
might become  
``
- Remember to modify all of your files including javascript and CSS which might have embedded links to static content. You can also move javascript and CSS files to the CDN as well, as long as they aren't dynamically generated.

## **Technique 1: Deliver static bits via Content Delivery Networks**

- **Why should this technique be used?**

- Most of your site is probably static content
  - Count up your images, style sheets, javascript files, media, and other blobs of static content. I'll bet it's 75% of your object count.
  - Think creatively about what "static" really means -- anything that gets served the same to everyone, even if it changes every 5 minutes -- could be served as a static object.
- Latency affects small objects in a disproportionate fashion
  - That 1k image will pay the same 100ms penalty to cross the Atlantic ocean that a 10 megabyte download will.
  - Many of the early page objects (especially CSS) download serially
- It's a fairly cheap fix
  - Plenty of competition means CDN costs compare favorable with what you're probably paying for bandwidth, plus you'll take load off your origins
- It's pretty easy
  - A little DNS and a few tweaks to your pages, and you're set.
  - Get results in hours or days, not weeks or months

## **Technique 1: Deliver static bits via Content Delivery Networks**

- **When is this technique inappropriate?**

- Content which cannot be cached
  - CDNs rely on use of a cache to deliver content locally.
  - Personalized content typically cannot be cached.
  - Ad delivery which requires cache busting also is inappropriate for CDNs.

## **Technique 2: Going Multi-site -- Challenges and Opportunities**

- **What does it mean when you go multi-site?**

- Typically, a web product is served from one or more machines in a single physical location. Multi-site delivery is served from geographically diverse locations, typically for high availability and performance reasons.
- Going multi-site requires design analysis to ensure that the application will continue to work as expected once split across multiple locations.
- Frequent problems include:
  - Applications which keep a session on front-end servers, requiring the use of “sticky” or “cookie-based” load balancing, as this is difficult to maintain with multiple locations.
  - Applications with extremely large, high-volume content repositories, where maintaining data consistency across multiple locations may be difficult.
  - Applications which do back-end communication for clustering via broadcast or anycast, as linking together multiple sites may be difficult.

## **Technique 2: Going Multi-site -- Challenges and Opportunities**

- **What do I do to get my application to run multi-site?**
  - Select appropriate geographically diverse locations to host your application.
    - Choose locations which can provide coverage with the latency you need
    - Network connectivity can be more important than physical geography
    - Consider diverse providers and networks to reduce risks
  - Deploy your application, making modifications as necessary
    - Strive to keep all locations as similar as possible (complexity drives cost)
    - Design to tolerate network interruptions between sites
    - Try to make the web tier as stateless as practical
  - Add a Global Server Load Balancing product to localize users
    - DNS-based GSLB with performance localization (not just Round Robin)
    - Available in some form on most every switch vendor (F5, Citrix, Cisco, etc)
    - Also offered as a service (UltraDNS, CDN providers, etc)
    - Make your GSLB servers the DNS authority for your distributed sites

## **Technique 2: Going Multi-site -- Challenges and Opportunities**

- **Why should this technique be used?**

- Going multi-site gets your entire product localized, dynamic and static
  - Some applications are almost exclusively dynamic content
  - In other cases, significant localization is easiest with multiple sites
- Covering certain regions requires a physical presence (legal reasons)
  - Privacy/data retention laws (European Union)
  - Special network filtering requirements (China)
- Building multiple, redundant sites reduces provider risk
  - Depend on multiple unrelated backbones and power grids
  - Protect against provider disputes, financial instability, growing pains
  - Avoid undue impact due to natural and man-made disasters

## **Technique 2: Going Multi-site -- Challenges and Opportunities**

- **When is this technique inappropriate?**

- Cost
  - Building more sites will probably cost more money
  - Geographically diverse sites will need some local staff
- Application Design
  - Some applications just don't work well when distributed
  - A few applications don't benefit from lower latency

 **Thanks!**

**Questions?**

email: [jacob@aol.com](mailto:jacob@aol.com)