# CS 591-06, Lecture 11

Jared Saia
University of New Mexico

## Class Outline

*A reasonable probability is the only certainty.* - E.W. Howe

- Review Chernoff Bounds
- In-Class Exercise
- The Akamai Problem
- The Akamai Solution



## Chernoff Bounds

- Let $X = \sum_{i=1}^{m} X_i$ where the $X_i$ are random variables which are: independent and all take on values between 0 and 1.

then for any $0 < \delta \leq 1$,

- $P(|X - E(X)| \geq \delta E(X)) \leq 2e^{\frac{-E(X)\delta^2}{4}}$



## Some Useful Bounds

- Stirlings Formula: $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \frac{1}{12n} + O(\frac{1}{n^2})\right)$
- $n! \geq \left(\frac{n}{e}\right)^n$
- $\binom{n}{k} \leq \frac{n^k}{k!}$
- $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$

## In Class Exercise

- Consider a directed random graph with possible loops on $n$ vertices where each of the $n^2$ possible edges is included with probability $p$.
- Q: What is the expected number of edges shared between any set of $n/2$ vertices?
- Q: What is the probability that some fixed set of $n/2$ vertices share more than $n^2p/4$ edges?
- Q: What is an upperbound on the probability that the graph contains *some* set of $n/2$ vertices which share $n^2p/4$ edges?



## The Congestion Problem

The WWW suffers from distribution delays. Two causes of delays:

- Congested Networks
- Swamped servers

Servers and networks can be swamped unexpectedly (e.g. "cool site of the day" phenomena)



## Solution to Congestion

- Caching is the solution to the congestion problem
- Nearby cache can serve a page quickly even if page source is swamped
- Good for the users and good for the group



## Naive Approach

- Naive Approach: Provide a group of users with a single shared cache machine
- Problem: if cache fails, all users are cut off
- Problem: a single cache can be a bottleneck
- Problem: Storage space of a single cache is limited so suffers false misses
- Problem: Limited number of users means limited amount of aggregated requests

## Cooperative Caching

- Cooperative Caching: Every client has one *primary cache*. If the request misses there, the client tries to locate the page in other cooperating caches.
- $+$ : fault tolerance, scalability and request aggregation
- - : consumes excess bandwidth and requires data duplication



## Akamai Approach

- Hash-based scheme - *consistent hashing*
- A user requesting a page contacts only the one cache responsible for that page, no inter-cache communication
- $+$ : less communication overhead per page request, discovers misses quickly, reliable
- - (?) : requires mathematical tools to develop the hashing scheme



## Hashing

- Goal: Let any client hash a URL to a particular cache containing that URL
- Naive Approach: Hash URL u to cache via function $h(u) = 7u + 4(mod 23)$
- Problem1: What happens when a 24th cache is added to the system? (If we change to new function $h'(u) = 7u + 4(mod 24)$, then essentially every URL is mapped to a different cache.)
- Problem2: What happens if all of the clients have different ideas of which servers are available