Mini-Flash Crowds

Balachander Krishnamurthy http://www.research.att.com/~bala/papers

Joint work with

Pratap Ramamurthy and Aditya Akella (Univ of Wisconsin)

Vyas Sekar (CMU), Anees Shaikh (IBM Research)

Inferring resource constraints of remote Web servers

Motivation

- Identify resource constraints in infrastructure
- Site operators can test ability to withstand real load
- Identify specific resources that are taxed
- Improve infrastructure against simultaneous legitimate requests (this is not DDoS mitigation work)

Flash Crowds

- 1971 Larry Niven science fiction short story, many people could teleport to see historical events anew.
- Too many people wanted to go to the same day - flash crowd
- Victoria Secret webcast, WCS, Olympics – but one can provision for these
- Slashdot effect
- On social networks: "samy is my hero" XSS worm (via AJAX + js added his profile to 1M other myspace users)

Mini-Flash Crowds

- What if you are able to cause an increase in load on a site...
- ...without it being appreciable (i.e., site does not notice it)
- Yet it is possible to

  Understand the response curve
  Infer bottlenecks
  Estimate possible tipping point

What is a MFC?

- A set of controlled measurements
- From a steadily increasing number of clients (with limit)
- *Synchronized* requests to server being tested
- Various request types to exercise

  Network bandwidth
  Local disk
  CPU
  Back-end database

- Other resources can also be tested

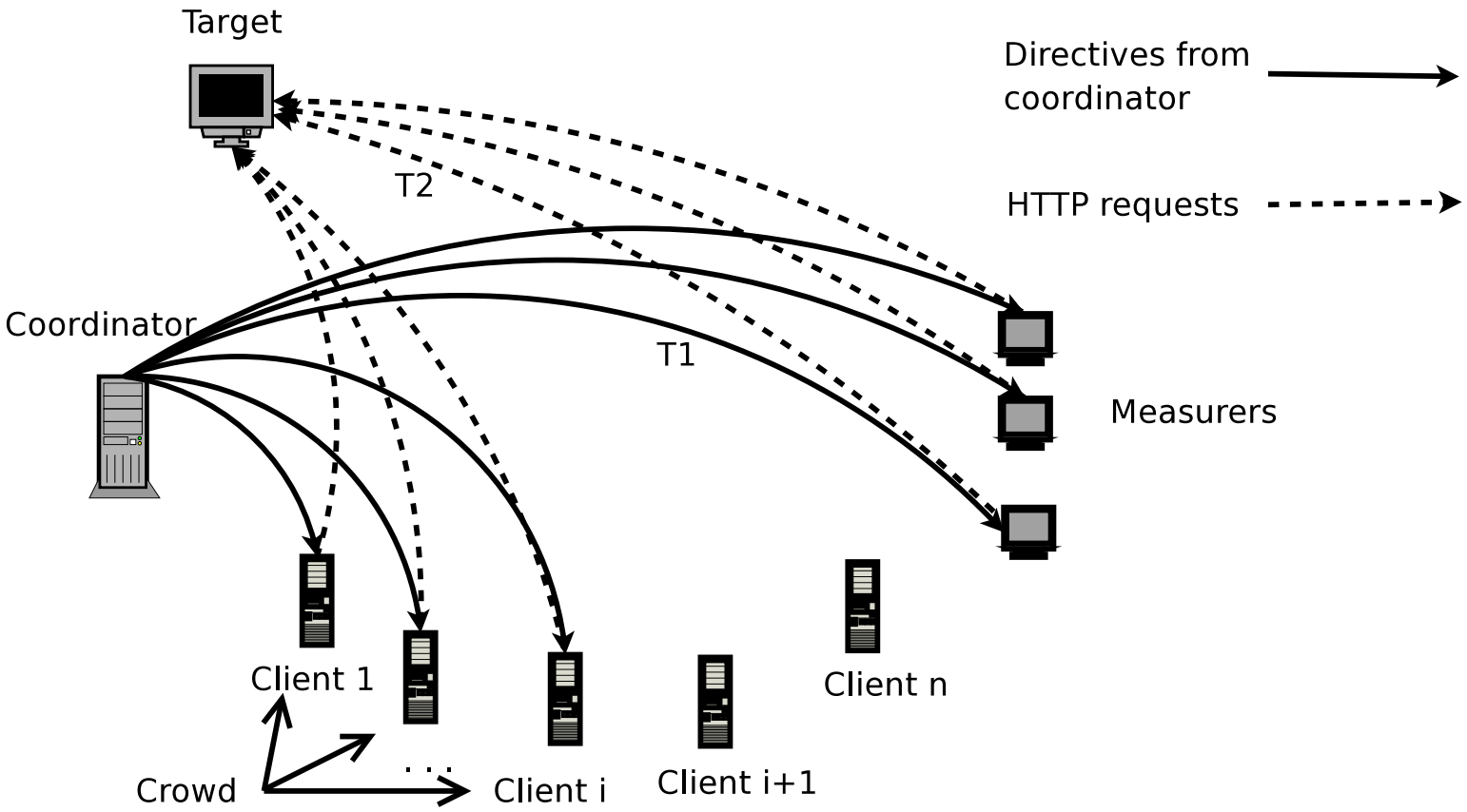
Key insight

- Watch for *small* but *discernible* increase in response time
- Slow but steady increase in # of clients (and simultaneous requests made)
- Initial response time increase threshold
- If increase noticed, we stop
- If maximum number of clients is reached *without* increase, we stop

Key advantages

- Light-weight experiment setup
- Non-intrusive wrt server (we lose if we are detected)
- No involvement from production servers (if available, we can do better)
- Real/distributed set of clients: reflects wide-area network conditions

MFC experiment structure



Experiment flow: Profiling stage

- Crawl subset of site, classify objects (html, binary, images, queries..)
- Obtain meta-information (e.g., size) via HEAD and categorize into small objects ($<$10KB), large objects ($>$100KB)
- Use GET for small queries (also $<$10KB)

Experiment flow: Object and base stages

- Initial list of clients send synchronized requests
- Requests vary with object type
- Unique small object and small query (when available) impacting disk and database back-end
- $Same$ large object impacting access bandwidth
- Base stage: HEAD request for index.html - baseline for request processing

Validation: Identifying Resource constraints

Aim: Narrow down impact on specific server resources

- Clients on LAN, local server (Apache 2.2, Ubuntu Edgy 2.6.* kernel)
- *atop* used to monitor server resources: CPU, memory, disk accesses, network usage. Crowd sizes from 15-50 with increments of 5.
- *same* small objects: Disk caching seen
- *unique* small objects: 20x response time increase (more disk reads)
- 100KB large object: Network bandwidth constraint increases response time significantly
- Backend database used to examine same and unique queries (with both FastCGI and Mongrel interfaces) - unique showed higher CPU utilization

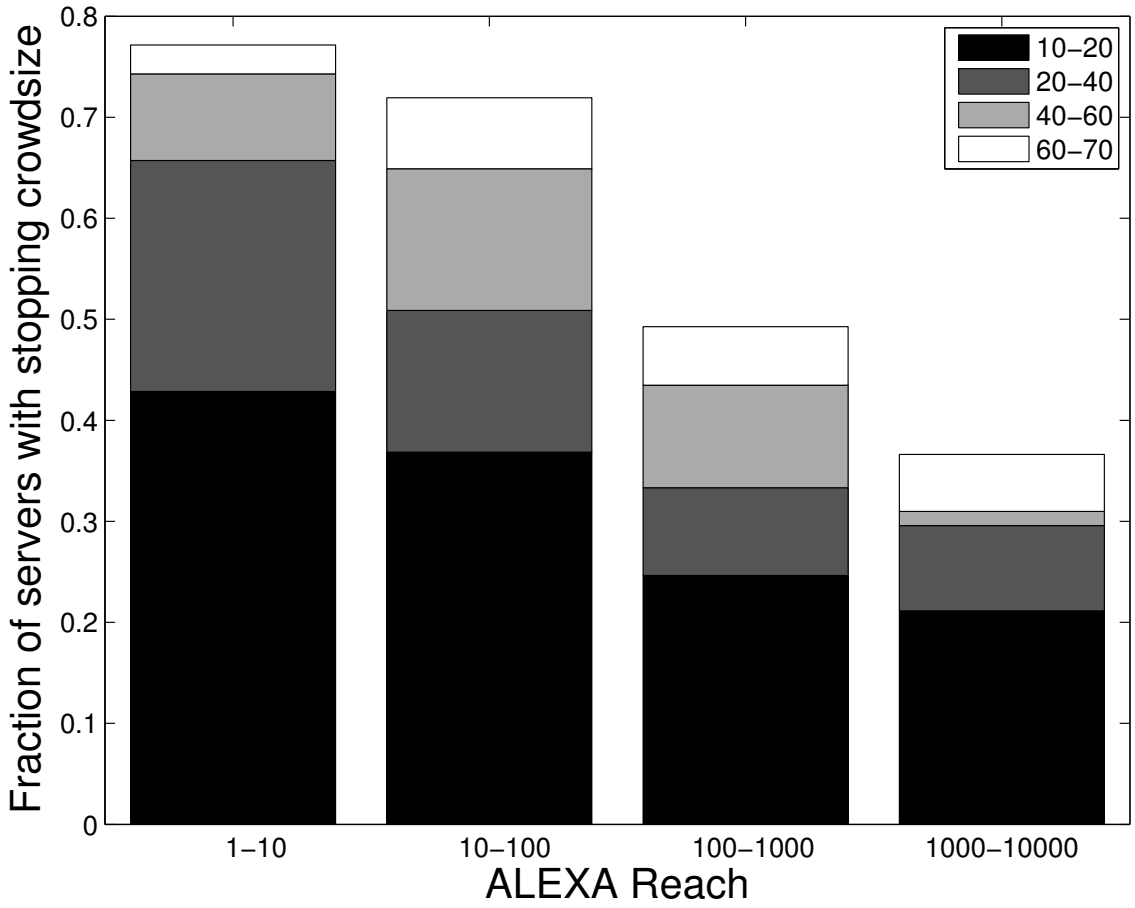
Wide area experiment-1: Web sites

- Base stage of MFC (HEAD request) over 1 week from 65 P'lab clients
- Against 200 live Web servers
- Stopping threshold 100ms
- Small enough to be not very intrusive at a site
- Large enough for website to worry human-perceived interaction time
- Threshold could be a function of the base response time and type of experiment (large vs small object) etc.

Wide area experiment-1: Web sites (continued)

- Sites grouped into categories based on Alexa reach-per-million (rpm of 1000 means 1000 out of 1 million users visited it)
- Crowd size at which degradation of $>$ 100ms response time seen, broken down by crowd size values into sub-ranges
- Larger reach categories show smaller fraction of servers that degrade
- Surprisingly $>$ 30% of sites even in 1000-10000 rpm range show degradation with $<$ 65 simultaneous requests.

Stopping crowd sizes for various rpm with HEAD request



Wide area experiment-2: Phishing Sites

- Curious how such sites are provisioned; small study
- 44 different phishing sites, 40 showed a 100ms response time increase with a crowd size less < 45, 27 with crowd size < 15
- Compared to Web servers in rpm 1-10, fraction of sites was 40%
- Most phishing sites are hosted on low-end servers as one would expect

Discussion

- Differences with real flash-crowds: controlled setup, so requests look normal. We ensure no sudden surge.
- Limitation: strictly response-time increase based and thus black-box
- Assumption: server load increase monotonically with crowd size. Not true if server caches objects, multiple replicated servers are used, dynamically re-provisions – so absence of response time increase can be inferred as well-provisioned.

Discussion - continued

- Multi-server Websites: We assume single IP address/single host - invalid for sites using load balancing, CDNs. MFC cannot handle reactive load balancing techniques well, yet.
- Security Implications: Parameters chosen to ensure non-intrusiveness
- Implementation inefficiencies vs. Performance prediction: We want to identify implementation inefficiencies/resource bottlenecks and provide a framework for site admins to *predict* performance under load. We can do former; latter goal requires providing a full load-response curve.
- Implications for Administrators: Need meaningful suggestions for network operators.