Paul Howe
Prof. Brooks
CS680
May 9, 2010

# HCKRN.WS Discussion

hckrn.ws [1] is a mobile front end for the link aggregator, Hacker News (news.ycombinator.com) [2]. I'll refer to hckrn.ws as HNM (HN Mobile) for short for the rest of the discussion. HNM is currently being hosted on a small VPS [3] instance which includes 256M of memory and one virtual quad-core AMD Opteron processor. In general my goals were the following:

- Determine the feasibility of hosting a performant web application in a relatively resource limited environment
- Understand the practical implications of writing event based servers
- Gain an intuition for writing event based servers to inform an opinion in the thread vs. event scalability debate

## System Details & Architecture

HNM is an evented application built on node.js, an asynchronous Javascript server framework which utilizes the V8 Javascript engine [4] for JS fast interpretation. Node.js is designed to support a highly asynchronous programming approach through callbacks. This approach, though awkward in many languages, is a fairly good fit for Javascript thanks to its support for anonymous functions and closures.

```
// attach periodic event to event loop
// refresh feed info
setInterval(function(){

  // removed some decls for readability

  req.addListener('response', function (res) {

    // stream in feed data
    var jsonData = '';
    res.setEncoding('utf8');
    res.addListener('data', function (chunk) {
      jsonData += chunk;
    });

    // when finished process feed
    res.addListener('end', function(){

      var value = eval('(' + jsonData +')');

      try{
        // process feed data
        cradleDb.insert(newDocs, function(err, res){
          // update caches after completion of insert
          linksIndex = utils.getLinks(cradleDb);
        });

      }catch(TypeError){ /* ignore */ }
    });
  });
  req.end();
}, RSS_REFRESH_INTERVAL);
```

Fig 1

In the example above, notice that flow moves down to the bottom of the page but also to the right as higher nesting indicates code that is running later in the ordering scheme. The most highly nested section (following the database insert) is running at the end of a quadruply nested anonymous function chain. This type of nesting is necessary from the application developers perspective in order to insure certain ordering goals. For the Node event loop though this type of flow decoupling is necessary to allow Node to service other events while the blocking portions are in a waiting state.

The HNM application server is responsible for gathering updated feed information, parsing it, inserting it into the database, updating cache information and responding to http client requests. Node supports trivially attaching events to its running event loop with a Javascript setInterval function which takes a function object and an interval value. Node then attaches this function to the event loop and periodically runs it on the specified interval. This type of construct is handy for many types of applications which need to run some piece of code periodically without requiring additional tools (such as cron). In this case, the HNM server periodically checks for feed updates and if it finds any incorporates them into its local database and updates its serving data cache.

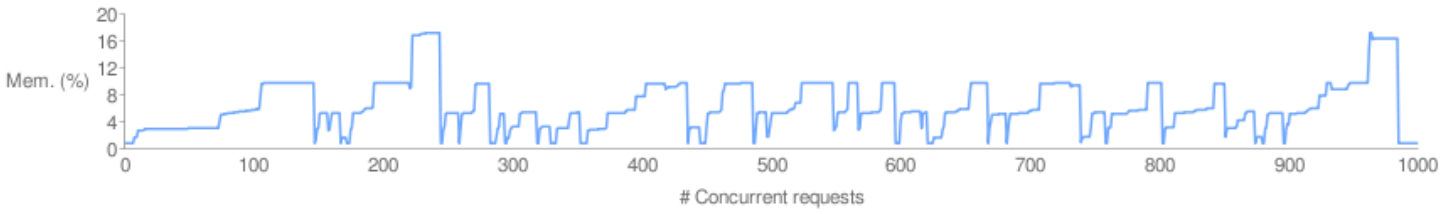
HNM uses CouchDB for persisting data between runs. Couch refers to database objects as documents which are simple Javascript object hashes. Though written in Erlang, Couch is a good fit for Node projects because all of its data objects are stored as JSON serialized data structures. In addition to the documents as object hash model, Couch queries ('views' in Couch parlance), are really anonymous Javascript functions which map an input to an output set. Couch has a variety of interesting other features though they are outside the scope of this discussion.

## Performance

Though HNM is a simple application, I believe its performance on a relatively minimally resourced host demonstrates the effectiveness of evented programming. My first goal was to quantify the general responsiveness of the application with testing tools such as Apache Bench. At high concurrency rates the application was able to server around 330 reqs/sec (raw output can be found in /data).

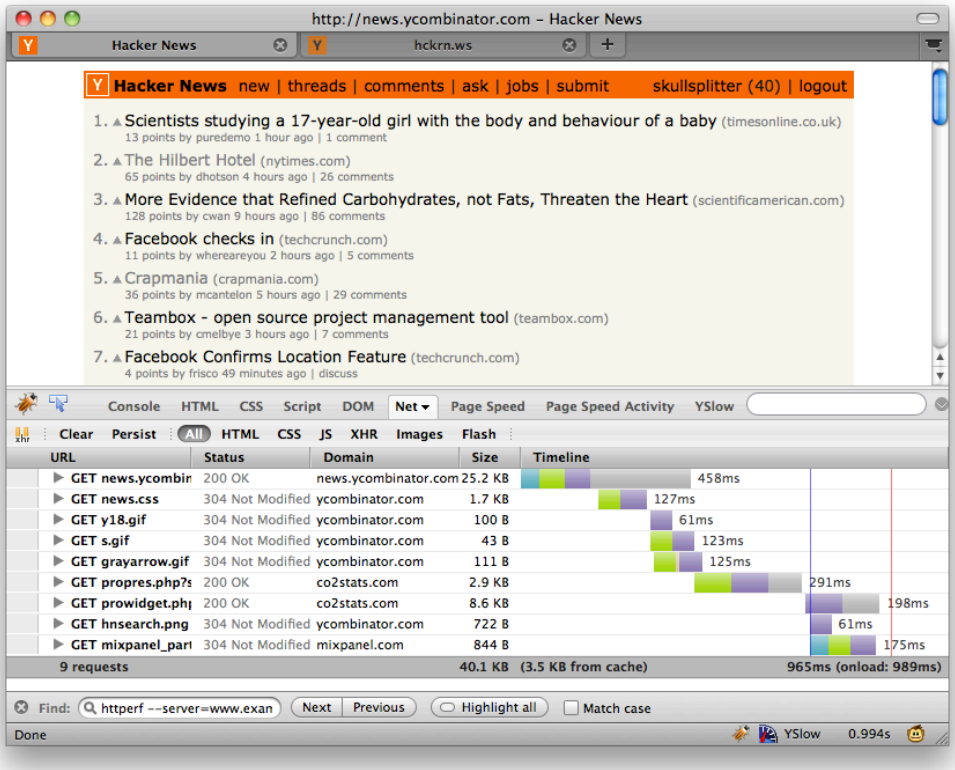
Though response times are important, one of the main arguments for event based programming is for their supposed smaller memory demands under concurrent load. Figure 2 shows a graph of memory usage over time for the application server (executed as 'node hn.js'). Gathering exact memory usage vs # of concurrent clients is tricky because it requires coordination across machines (memory usage is available on the server, concurrency amount is available on the testing machine). The other note on the following graph is that the x-labels are confusing. The memory usage represents a series of AB runs each for 1000 reqs. at varying levels of concurrency. For example, the first run was for 1000 requests for 100 clients, the second for 200 clients etc. This creates a jagged plot as the server 'warms' up during each run of 1000 requests. Importantly though we notice several things about the memory usage. The first is that it seems more or less bounded. Memory usage increases to a point but levels off. I don't believe that this behavior would scale similarly forever as we are probably bumping

Fig 2



up against some system bottleneck (probably in the OS or the web server). However, its comforting to note that even under high load the server consumes only a modest amount of memory (at all times less than 20%).

The last set of performance measurements we will consider are from the the browsers perspective. Comparing the browser timing results from loading and rendering news.ycombinator.org and hckrn.ws respectively, hckrn.ws (HNM) compares favorably. Though this is arguably an unfair comparison since isolating the differences between each application is not practical, I still believe its compelling in that it gives a somewhat more intuitive feeling for the relative performance of HNM.

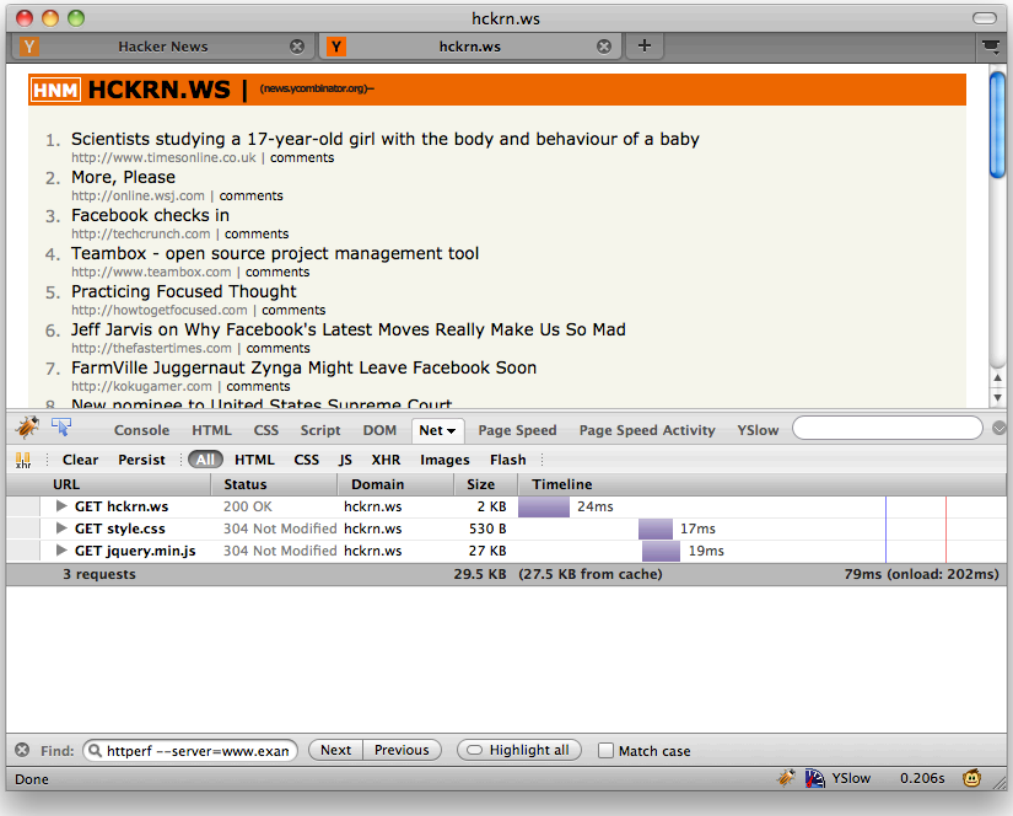



news.ycombinator.org

news.ycombinator.org is shown here loading in roughly 1000ms. This is perhaps on the slow side for a relatively high profile public site, on the other hand its still quite usable. The site requires 9 requests to load all assets after which it is able to fire the Javascript `onLoad` event. Another way to think about the response time is to consider only the main data request and none of the other page assets (images, css, js etc). The main data response returns in 458ms, the rest of the loading period is filled fetching the referenced assets. HNM includes very few assets so one could argue that this is a fairer timing to be used for comparison.

In any case, lets turn to HNM. HNM is composed of only three assets (data payload, a stylesheet and a Javascript library). `onLoad` fires at about 200ms but interestingly the real download time only takes ~80ms. As I stated earlier, I don't wish to read too much into these results but I include them here for comparison sake to give a sense of performance at the browser level. The bottom line of this comparison is that as a consumer service, HNM should be eminently usable on devices running over high latency mobile networks.

While not strictly a performance attribute, regarding the stability of the system I was pleasantly surprised. Node is a spartan framework but it appears to be sufficiently robust. The application remained up through each AB run (including a heavy run of 100k requests). And while the server showed some signs of sluggishness during these tests it nevertheless managed to remain alive and able to server requests.




HNM

## Events vs. Threads

There's been an increasing amount of interest in the threads vs. events serving style debate. Why Events Are A Bad Idea [5] argues that the threading model is a better abstraction for writing concurrent applications in that it better fits the programmers mental model of the problem space. They also argue that the perception that thread based servers have higher memory requirements is mainly due to side effects of current implementations. The authors develop their own thread based servers which they argue have the same scaling properties as event loop servers.

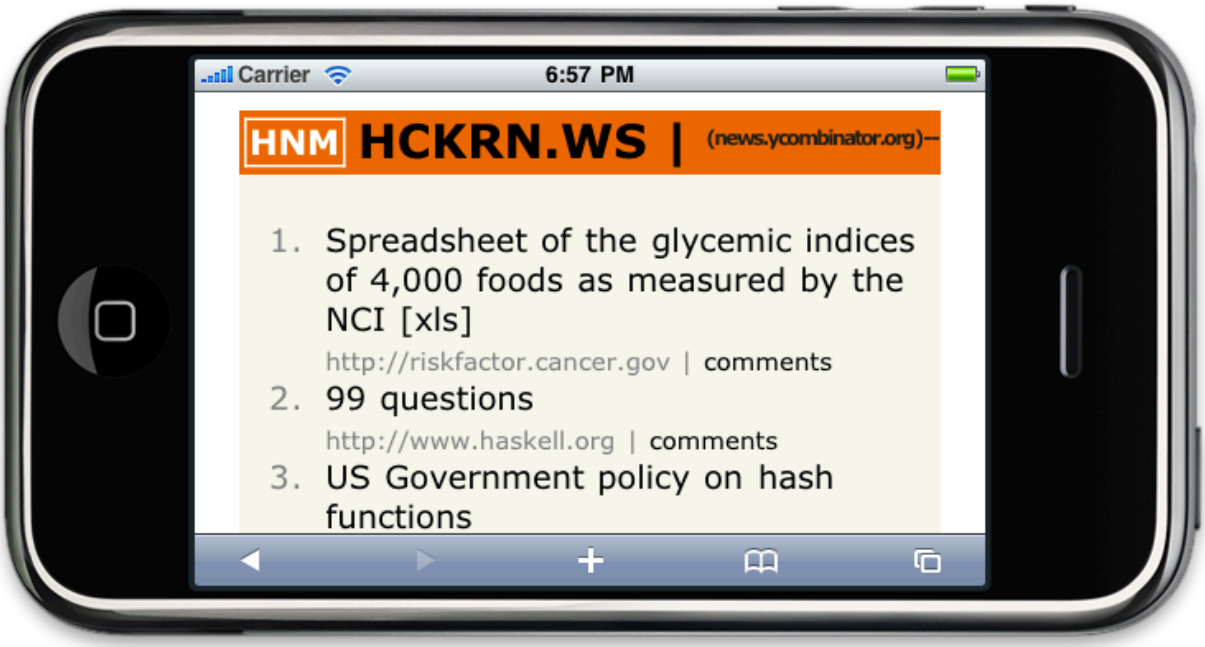
Ideally, if the threading library improvements perform as stated we can hope to see a resurgence in interest to using thread based servers to support applications which handle many concurrent clients.

As to the argument that threading is a better abstraction, I believe the jury is still out. The threading based model has enjoyed a long history of dominance and this has led to a large amount of support with respect to tools and conceptual momentum. Are event based programs less comprehensible to the average programmer? Perhaps but I believe that as a generation of new developers who cut their teeth on client side web programming emerge this will change. UI programming is naturally event based (`onClick`, `onSubmit` etc) and as familiarity in this style of programming increases, so does the attraction to event-based programming models offered by tools such as Node. The skills needed to

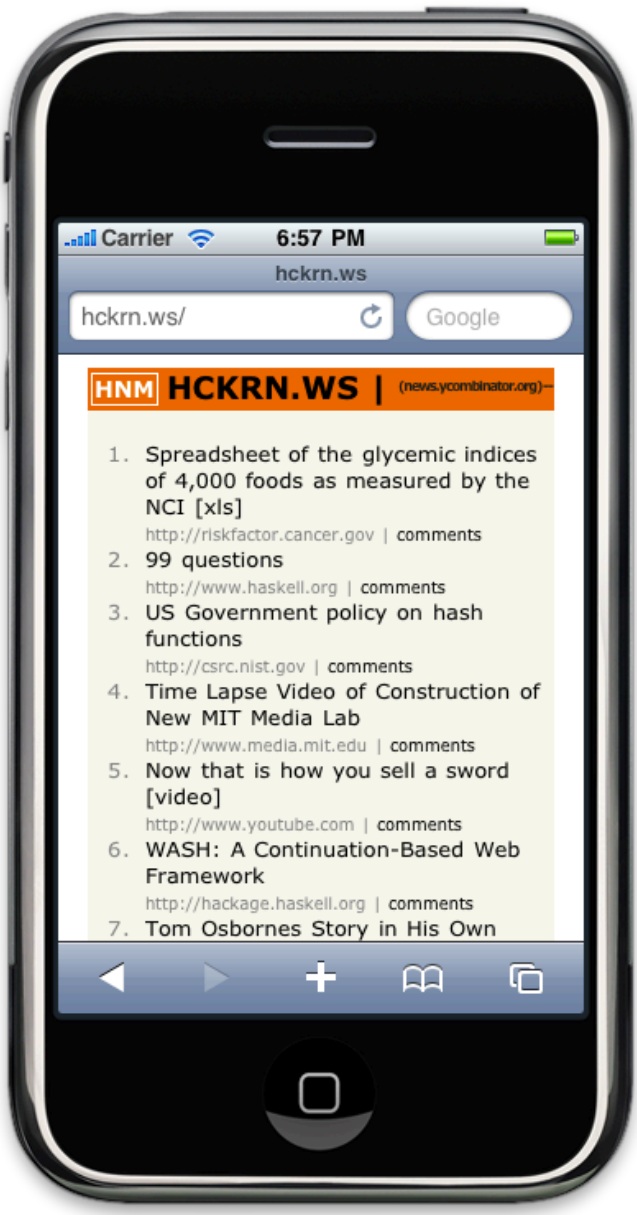
write a successful Javascript driven web interface are remarkably similar to those needed to write an event based server. Though the threading model may feel like a more natural abstraction today, I have a sense that there is a growing amount of familiarity for programming with events which is shifting the balance.

## User Interface Comparison

The following figures are included to show the improvement in readability on a mobile device. Many UI elements were considered to improve the display on smaller devices including font faces, viewport dimensions, element placement etc.


Carrier
6:57 PM
HNM HCKRN.WS | (news.ycombinator.org)--
1. Spreadsheet of the glycemic indices of 4,000 foods as measured by the NCI [xls]
http://riskfactor.cancer.gov | comments
2. 99 questions
http://www.haskell.org | comments
3. US Government policy on hash functions


Carrier
6:58 PM
Hacker News
news.ycombinator.com/
Google
Hacker News new | comments | ask | jobs | submit
login
1. More Evidence that Refined Carbohydrates, not Fats, Threaten the Heart (scientificamerican.com)
68 points by cwan 3 hours ago | 44 comments
2. Charlie Stross on Common Misconceptions About Publishing: Ebooks (antipope.org)
51 points by bensummers 4 hours ago | 5 comments
3. Reddit IAMA. An ex Google programmer switched to a job in the lumbering industry (reddit.com)
95 points by tfh 7 hours ago | 51 comments
4. How (and when) to motivate yourself (hbr.org)
46 points by cŏriander 5 hours ago | 6 comments
5. Many gifted children fail academically (umontreal.ca)
38 points by tokenadult 3 hours ago | 43 comments
6. New Programming Jargon (globalnerdy.com)
43 points by tortilla 5 hours ago | 17 comments
7. The Productivity Myth (hbr.org)


Carrier
6:57 PM
hckrn.ws
hckrn.ws/
Google
HNM HCKRN.WS | (news.ycombinator.org)
1. Spreadsheet of the glycemic indices of 4,000 foods as measured by the NCI [xls]
http://riskfactor.cancer.gov | comments
2. 99 questions
http://www.haskell.org | comments
3. US Government policy on hash functions
http://csrc.nist.gov | comments
4. Time Lapse Video of Construction of New MIT Media Lab
http://www.media.mit.edu | comments
5. Now that is how you sell a sword [video]
http://www.youtube.com | comments
6. WASH: A Continuation-Based Web Framework
http://hackage.haskell.org | comments
7. Tom Osbornes Story in His Own


Carrier
6:58 PM
Hacker News
news.ycombinator.com/
Google
Hacker News new | comments | ask | jobs | submit
More Evidence that Refined Carbohydrates, not Fats, Threaten the Heart (scientificamerican.com)
Charlie Stross on Common Misconceptions About Publishing: Ebooks (antipope.org)
Reddit IAMA. An ex Google programmer switched to a job in the lumbering industry (reddit.com)
How (and when) to motivate yourself (hbr.org)
Many gifted children fail academically (umontreal.ca)
New Programming Jargon (globalnerdy.com)
The Productivity Myth (hbr.org)
What's the deal with these weird comments on my blog? (thepursuitofquality.com)
Picwing is Offering 2 Months Free in Honor of Mother's Day (ycombinator.posterous.com)
Ongoing Facebook API Outage Worries Developers. Can the Web Depend on FB? (nytimes.com)
Time Lapse Video of Construction of New MIT Media Lab (mit.edu)
Spreadsheet of the glycemic indices of 4,000 foods as measured by the NCI [xls] (cancer.gov)
What Business is Wall Street In? (blogmaverick.com)
Bedbugs in the Duvet: An infestation on the Upper East Side (nymag.com)
'Zombie' satellite runs amok in Earth's orbit (csmonitor.com)
Paul Carr on Facebook Privacy (techcrunch.com)
WASH: A Continuation-Based Web Framework (haskell.org)
99 questions (haskell.org)
Linux and Seeds, Geeks and Farmers (a42.com)

## Bibliography


1. HCKN.WS http://hckrn.ws
2. Hacker News http://news.ycombinator.com
3. PRGMR http://prgmr.com/xen/
4. V8 http://code.google.com/p/v8/
5. von Behren, R., Condit, J., and Brewer, E. 2003. Why events are a bad idea (for high-concurrency servers). In Proceedings of the 9th Conference on Hot Topics in Operating Systems - Volume 9 (Lihue, Hawaii, May 18 - 21, 2003). USENIX Association, Berkeley, CA, 4-4.
6. Are Events Fast? http://www.pps.jussieu.fr/~jch/research/cpc-bench.pdf