A Conversation with Matt Wells

PHOTOGRAPH BY DEBORAH MANDEVILLE

Search is a small but intensely competitive segment of the industry, dominated for the past few years by Google. But Google's position as king of the hill is not insurmountable, says Gigablast's Matt Wells, and he intends to take his product to the top.

Wells began writing the Gigablast search engine from scratch in C++ about three years ago. He says he built it from day one to be the cheapest, most scalable search engine on the market, able to index billions of Web pages and serve thousands of queries per second at very low cost. Another major feature of Gigablast is its ability to index Web pages almost instantly, Wells explains. The network machines give priority to query traffic, but when they're not answering questions, they spend their time

When it comes

TO COMPETING IN THE
SEARCH ENGINE ARENA, IS
BIGGER ALWAYS BETTER?

spidering the Web.

Wells got his start in search in 1997 when he went to work for the search team at Infoseek in Sunny-

vale, California. He stayed there for three years before returning to New Mexico and creating Gigablast. He has a B.S. in computer science and M.S. in mathematics from New Mexico Tech.

While at Infoseek, Wells came to know Steve Kirsch, who founded and led that Internet navigation service before its acquisition in 1999 by Disney. Kirsch is now the founder and CEO of Propel Software, which develops acceleration software for the Internet. A seasoned entre-

preneur, he started Frame Technology in 1986 and Mouse Systems in 1982. As a result of his business successes, *Red Herring* included him on its Top Ten Entrepreneurs of 2000 list, *Newsweek* called him one of the 50 most influential people in cyberspace, and most recently, *Forbes* named him to its Midas List.

Kirsch has been involved with the Internet since 1972, when he worked at the first Arpanet node as a systems programmer while attending high school. He has B.S. and M.S. degrees in electrical engineering and computer science from MIT. He has established a \$50 million charitable foundation, as well as the Center for Innovative Policies to pursue his commitment to sustainable energy and for educational reform.

Recently Kirsch sat down with his former employee, Wells, to discuss search—its evolution and future.

STEVE KIRSCH Matt, tell us a little bit about your background. How did you get into search?

MATT WELLS I got started in search while pursuing my graduate degree in mathematics at New Mexico Tech, a small technical school in the middle of the desert. There I designed and implemented the Artists' Den for Dr. Jean-Louis Lassez, the department chairman of computer science. It was an open Internet database of art. Anyone could add pictures and descriptions of their artwork into the searchable database. It was a big success and led me to my first "real" job in 1997, working on the core team at Infoseek. Shortly after Infoseek was acquired by Disney I started working on Gigablast, my own search engine.

SK What problems do search engines face today?

MW There are many. Search was quoted by Microsoft as being the hardest problem in computer science today, and they weren't kidding. You can break it down into three primary arenas: (1) scalability and performance; (2) quality control; and (3) research and development.

SK Why is scalability an issue?

MW Scalability is a problem for a few reasons. For one, an industrial-grade engine has to be distributed across many machines in a network. Having thousands and even tens of thousands of machines is not unheard of. When you deal with that much data you have to be concerned with data integrity. You need to implement measures to compensate for mechanical failure. You would be surprised at the amount of data corruption and machine failure in a large network. Today's systems also run hot, so a good cooling system will help reduce failures, too.

Secondly, everything an engine does has to be distributed uniformly across the network. The simplest function of adding a record to a database is a lot more complicated

when you are in a distributed environment. You have to identify the set of hosts that should store that record, ensure they are up, transmit it to them as fast as possible, and gracefully handle any error condition that might arise.

Thirdly, the sheer amount of data in the index has to be processed as fast as possible to ensure minimum query times and maximum spider rates.

SK What's at the heart of a query?

MW At the core of any search engine is the index. The fundamental data structure of the index is called a termlist, posting list, or index list—the terminology is different depending on where you are. The termlist is essentially a list of document identifiers for a given term. For example, the term *cat* might have doc IDs 1, 5, and 7 in its termlist. That means that the word *cat* appears in documents 1, 5, and 7. So when someone does a search for *cat*, you can tell them where it is.

The tricky part is when you search for *cat dog*. Now you have to intersect two termlists to find the resulting doc IDs. And when there are 20 million documents that have *cat* and another 20 million that have *dog*, you have quite a task ahead of you.

The most common way of dealing with this problem is to divide and conquer. You must split the termlist's data across multiple machines. In this case, we could split our search engine across 20 machines, so each machine has about 1 million doc IDs for *cat* and *dog*. Then the 20 machines can work in parallel to compute their top 10 local doc IDs, which can be sent to a centralized server that merges the 20 disjointed, local sets into the final 10 results.

There are many other shortcuts, too. Often, one of the termlists is sparse. If this happens, and your search engine is a default AND engine, you can b-step through the longer termlist looking for doc IDs that are in the shorter termlist. This approach can save a lot of CPU.

To add chaos to confusion, a good engine will also deal with phrases. A popular method for doing phrase searches is to store the position of the word in the document in the termlist as well. So when you see that a particular doc ID is in the *cat* and the *dog* list, you can check to see how close the occurrences of *cat* are to *dog*. Documents that have the words closer together should be given a boost in the results, but not too much of a boost or you will be *phrase heavy*, like AltaVista is for the query: *boots in the uk*.

SK What about stop words?

MW Another popular shortcut is to ignore stop words, like *the*, *and*, and *a*. Not having to scan through these really long termlists saves a considerable amount of CPU burn time. Most engines will search for them only if you explicitly put a plus (+) sign in front.

Yet another speed-up technique is to use caches. Reading data from disk, especially IDE [integrated drive electronics], is very slow. You want to have as many termlists crammed into memory as possible. When you have tens of thousands of machines, each with a gig or more of RAM, you have so much RAM that you can fit all of your index in there and disk speed is no longer an issue, except when looking up *title records*.

SK Can you describe a title record?

MW Every doc ID has a corresponding title record, which typically contains a title, URL, and summary of the Web page for that doc ID. Nowadays, most engines will contain a full, zlib-compressed version of the Web page in the title record. This allows them to dynamically generate summaries that contain the query terms. This is very useful. Note, that each search engine probably has its own word for *title record*; there's no standard terminology to my knowledge.

SK You mentioned three general problem areas for search. How about the second one, quality control, as it pertains to search?

MW Quality control is about enhancing the good results and filtering out the bad. A good algorithm needs to analyze the neighboring link structure of a Web page to assign it an overall page score with which to weight all the contained terms. You also need to weight the terms in the query appropriately. For instance, for the query, *the cargo*, the word *the* is not as important as the word *cargo*. So a page that has *the* 100 times and *cargo* once should not outrank a page that has *cargo* a few times. Furthermore, you want to balance the weight of phrases in a query with the weight of single terms in the same query. You do not want to be too phrase heavy or too phrase light.

Filtering out the bad is a bit harder. You have to deal with hordes of people trying to spam your engine with millions of pages. You cannot employ any 100 percent automated system to remove these pages from the index. It requires a powerful set of tools that allow you to automate it as much as possible, but still retain some level of manual control.

SK I'm not surprised spam is still a big issue. Most of the audience is familiar with e-mail spam; can you describe

what spam is for a search engine?

MW Gladly. Search engine spam is not exactly the same as e-mail spam. Taken as a verb, when you spam a search engine, you are trying to manipulate the ranking algorithm in order to increase the rank of a particular page or set of pages. As a noun, it is any page whose rank is artificially inflated by such tactics.

Spam is a huge problem for search engines today. Nobody is accountable on the Internet and anybody can publish anything they want. A large percentage of the content on the Web is spam. The spammers are highly motivated to garner as many high-ranking positions as they can. It translates directly into more financial rewards.

One of the simplest ways to subvert a search engine, or at least try to, is to repeat keywords in the Web page. This tactic worked well in the mid-90s but not so much anymore.

One of the more common methods today is link spamming. Webmasters and SEOers [SEO stands for search engine optimization] exchange links with each other to fool the link analysis algorithms that all the big engines employ. Some of the more evil Webmasters will purchase hundreds of IPs supporting thousands of domains hosting millions of randomly generated pages. So you get this

entirely artificial Web community that boosts itself to the top of the results.

Each page has something like 1,000 random words. These guys are aiming for the tail end of the normal distribution. So anytime someone searches for a somewhat unpopular combination of terms, these spammers come up on top. Banning their IP addresses is not enough, because they move their domains around on a monthly basis.

SK Sounds like a major pain.

MW It is. Our own little slice of hell.

SK Any other quality issues?

MW Yes, there's more. Duplicate content. Mirrored sites. When you do a query, you don't want to see the same content listed a hundred times. In a lot of cases the sites in the search results will all be mirrors of each other. So a good engine will have mirror detection, or duplicate page removal algorithms that will excise the dupes from the search results. But if your duplicate detection parameters are too loose, you might end up removing important pages. It's a touchy subject.

Also, you now have businesses, such as Amazon and Reuters, encouraging people to incorporate their XML feeds into their Web pages. Everybody will dress the content up in different ways so that no two pages are exactly alike, but really—content-wise—they are the same and you'll expect a search engine to detect this and cluster the duplicates out of sight.

SK How do you specifically address some of these issues in Gigablast, such as taking the big intersection of doc IDs or dealing with spam?

MW I would really love to get into some of the other technical challenges and ways I address them through Gigablast, but if I go too far I may end up jeopardizing some of my trade secrets. When making public disclosures, I have to balance my technical side with my business side.

Search is a fiercely competitive arena, even though there are really only five Web search companies today: Google, Yahoo (Altavista/AlltheWeb/Inktomi), Looksmart (Wisenut), AskJeeves (Teoma), and Gigablast. It's a tight little community, and a lot of the people know and watch each other. Microsoft is also coming to the party, and everyone's a little bit nervous to see what it's bringing. When you run a search engine, all you have is the code. It's just software. So it's all about algorithms. You have to protect what you have, and the only way to do that is to keep your mouth shut sometimes.

SK What is the state of the search sector today?

MW Financially speaking, I'm no expert, but a lot of money is being generated in Web search—mostly from including context-relevant paid results with the actual search results. Enterprise search is just not as rich. You can tell this just by looking at the sizes of the companies in both spaces.

On the technical side of things, I would say that we are just beginning to realize the full potential of the massive data store on the Internet. Companies are struggling to sort and filter it all out.

SK I'm interested to know what you think of Google. Can you tell us something about that?

MW Google is definitely the one to beat. It has a near monopoly on the search market, but that's because it wisely focused on quality search results when everybody else was too busy turning into a portal and neglecting their search departments. Ahem.

SK Hey, it was the thing to do at the time.

MW Yes, well, at Infoseek a minuscule amount of the people who worked there actually worked on the search engine. I thought it was quite a bit unbalanced.

Getting back to the question, I would say Google's strength is its cached Web pages, index size, and search speed. Cached Web pages allowed Google to generate dynamic summaries that contain the most query terms. It gives greater perceived relevance. It allows searchers to determine if a document is one they'd like to view. The other engines are starting to jump on board here, though. I do not think Google's results are the best anymore, but the other engines really are not offering enough for searchers to switch. That, coupled with the fact that Google delivers very fast results, gives it a fairly strong position, but it's definitely not impenetrable.

I do not think Google's success is due that much to its PageRank algorithm. It certainly touts that as a big deal, but it's really just marketing hype. In fact, the idea predated Google in IBM's CLEVER project [a search engine developed at the IBM Almaden Computer Science Research Center], so it's not even Google's, after all.

PageRank is just a silly idea in practice, but it is beautiful mathematically. You start off with a simple idea, such as the quality of a page is the sum of the quality of the pages that link to it, times a scalar. This sets you up with finding the eigen vectors of a huge sparse matrix. And because it is so much work, Google appears not to be updating its PageRank values that much.

A lot of people think this way of analyzing links was invented by Google. It wasn't. As I said, IBM's CLEVER project did it first. Furthermore, it doesn't work. Well, it

works, but no better than the simple link and link text analysis methods employed by the other engines. I know this because we implemented our own version at Infoseek and didn't see a whole lot of difference. And Yahoo did a result comparison between Google and Inktomi before it purchased Inktomi and concluded the same thing.

What I think really brought Google into the spotlight was its index size, speed, and dynamically generated summaries. Those were Google's winning points and still are till this day, not PageRank.

SK Now tell us about Gigablast.

MW Gigablast is a search engine that I've been working on for about the last three years. I wrote it entirely from scratch in C++. The only external tool or library I use is the zlib compression library. It runs on eight desktop machines, each with four 160-GB IDE hard drives, two gigs of RAM, and one 2.6-GHz Intel processor. It can hold up to 320 million Web pages (on 5 TB), handle about 40 queries per second and spider about eight million pages per day. Currently it serves half a million queries per day to various clients, including some meta search engines and some pay-per-click engines. This, in addition to licensing my technology to other companies, provides me with a small income.

SK How did you come up with the name?

MW I sat down one day and made a list of "power" prefixes and suffixes. I had stuff like peta, mega, accu, drive, uber, etc. Almost all combinations I came up with were already owned in cyberspace, but gigablast.com was expiring soon, so I sniped it.

SK Why did you start Gigablast?

MW As you know, I worked at your old company,

Infoseek. After Disney snapped it up, it became a whole lot less focused on search than it already was and more focused on pretty-looking graphics and Web pages. I had a passion for search that could not be fulfilled there. I had a lot of ideas and no power to realize them. That's why I decided to start off on my own. I wanted to see how far I could run with it. And I love competition... but only if I win.

SK What did you learn from working at Infoseek?

MW I learned what is possible. Working there gave me a unique perspective in the information retrieval industry. There was quite a bit of talent there as well. I also got a chance to familiarize myself with the three problem areas I mentioned: scalability and performance; quality control; and research and development.

SK Data integrity is a big factor in large databases. How does Gigablast deal with the loss of a machine and with data corruption?

MW Gigablast employs a dual redundancy scheme. So, every machine in the network has a twin host. You can configure it to have as many twins as you want. The machines in the network continually ping each other, and if they detect that one machine is not replying promptly to its pings, then they label it dead. At that point nobody

will request any information from the dead host until it begins replying to his pings again. It's like poking someone to make sure they are awake before you ask them a question.

Gigablast detects corrupted data and patches the data automatically from its twin host. If the twin's data is corrupt as well, then the data is excised and discarded.

SK How do you plan to beat your competitors?

MW I'm hoping to build Gigablast up to 5 billion pages

this year. My income level should allow me to do that if I invest everything in hardware and bandwidth. It will still be a modest number of machines, too, not a whole lot more than the eight I have now. I am a firm believer that bigger is better. And faster, too. With more machines I will be able to serve up faster results.

I also plan to launch an experimental search system that will add a new twist to the way people search for information on the Web.

As for the licensing area, I'm only one guy, so I don't have any major expenses and can offer the best service, support, and price to my clients.

SK What do you see on the near-term horizon for search?

MW Search is very cutthroat so I don't like to go too far on the plank, for reasons I mentioned before. However, I think that search engines, if they index XML properly, will have a good shot at replacing SQL. If an engine can sort and constrain on particular XML fields, then it can be very useful. For instance, if you are spidering a bunch of e-commerce Web pages, you'd like to be able to do a search and sort and constrain by price, color, and other page-defined dimensions, right? Search engines today are not very number friendly, but I think that will be changing soon.

I also think the advancements of tomorrow will be based on the search engines of today. Today's search engines will serve as a core for higher-level operations. The more queries per second an engine can deliver, the higher the quality its search results will be. All of the power and information in today's engines have yet to be exploited. It is like a large untapped oil reservoir.

SK OK, Matt, now what is the really big picture?

MW To understand the direction search is taking, it helps to know what search is. Search is a natural function of the brain. What it all comes down to is that the brain itself is rather like a search engine. For instance, you might look out the window and see your car. Your brain subconsciously converts the image of your car into a search term, not too unlike the word *car* itself. The brain first consults the index in your short-term memory. That memory is the fast memory, the L1 cache so to speak. Events and thoughts that have occurred recently are indexed into your short-term memory, and, later, while you are sleeping, are sorted and merged into your long-term memory. This is why I think you sometimes have weird dreams. Thoughts from long-term memory are being pulled into short-term memory so they can be merged and then written back out to long term, and you sometimes notice

these thoughts while they are being sorted.

So, the brain has to keep everything in order. That way, you can do fast associations. So looking up the word *car*, the brain lets you know that your gas tank is empty. You were driving the car this afternoon, so that thought is fresh in your short-term memory and is therefore the first thought you have. A second later, after the brain has had a chance to pull some search results from slower, long-term memory, you are informed that your car insurance payment is due very soon. That thought is important and is stored in the cache section of your short-term memory for fast recall. You also assign it a high priority in your directive queue.

When the brain is performing a search, it must actually result in a list of doc IDs, just like a search engine. However, these doc IDs correspond to thoughts, not documents, so perhaps it would be better to call them *thought IDs*. Once you get your list of thought IDs, you can look up the associated thoughts. Thoughts would be the title records I discussed earlier.

The brain is continuously doing searches in the background. You can see one thing and it makes you think of something else very quickly because of the powerful search architecture in your head. You can do the same types of associations by going to a search engine and clicking on the related topics. Clicking on one topic yields another list of topics, just like one thought leads to another. In reality, a thought is not much more than a set of brain queries, all of which, when executed, will lead to more thoughts.

Now that the Internet is very large, it makes for some well-developed memory. I would suppose that the amount of information stored on the Internet is around the level of the adult human brain. Now we just need some higher-order functionality to really take advantage of it. At one point we may even discover the protocol used in the brain and extend it with an interface to an Internet search engine.

All of this is the main reason I'm working with search now. I see the close parallel between the search engine and the human mind. Working on search gives us insights into how we function on the intellectual level. The ultimate goal of computer science is to create a machine that thinks like we do, and that machine will have a search engine at its core, just like our brain. Q

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

© 2004 ACM 1542-7730/04/0400 \$5.00