;login:

THE MAGAZINE OF USENIX & SAGE

April 2002 • Volume 27 • Number 2

inside:

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

USENIX news

The Long Wave

by Daniel Geer

President, USENIX Board of Directors

geer@usenix.org

A high percentage of the lay press and people in general have some idea that computer horsepower keeps getting better, i.e., they've heard of Moore's law and its halving of price per unit performance every 18 months. Economists, such as the American Economic Association's Dale Jorgensen, commentators, such as the *American Spectator*'s George Gilder, and senior government officials, such as Senator Ron Wyden, each in their own way point to the interaction of tax law and Moore's law as *the* source of wealth expansion over the last decade. Who am I not to concur?

But Moore's law has another important aspect – it predicts the future. Predicting the future well is a great source of capital, monetary and intellectual, as well as a guide on how to spend your monetary and/or intellectual capital. Thinking of Moore's law as a "curve," I'd like to add two other curves to the mix and hazard a prediction or two. Both curves are, like Moore's, more approximations based on observation rather than laws of physics. All three are imperfect, but let's ignore that imperfection for the moment.

The first curve is the price of storage. For the purpose of this article, it shows a similar form to the curve for computing, i.e., a halving in price per unit volume every cycle where, in this case, the cycle is 12 months rather than 18. An approximation to be sure, but a well-studied approximation (see, e.g., Clayton Christensen's *The Innovator's Dilemma* for a book-length treatment).

The second curve is the price of bandwidth – raw communications bandwidth. Again for the purpose of this article, let's say it, too, shows a halving in price per unit volume every cycle – in this case, 9 months rather than 12 or 18. Another approximation to be sure, and one that in a full econometric model would have to be adjusted to take into account both regulatory and monopoly supply issues. For the moment, though, I am talking just about raw costs – system inputs, in other words.

Now even if the numbers 18, 12, and 9 are off, these power curves have a relationship to each other that is at least ordinal, i.e., computing horsepower increases pretty awesomely but for storage it is even more awesome while for bandwidth it is more awesome yet. So what might it mean if we had, say, another decade of increases in bang for the buck along each of these fundamental axes?

Taken over a decade, computing's cost effectiveness would increase by a factor of 100, storage by 1000, and networking would be 10,000 times more cost effective. What I am interested in, however, is not these raw numbers but the relative change in what a networked computer might be like 10 years from now as compared to today. Within a decade, the CPU would be facing 10 times as much data per available cycle and 100 times as much bandwidth; yet draining the entire storage of such a machine over the net would take only one-tenth the time, even though the overall data volume had gone up three orders of magnitude. Putting it differently, this is not the world with which we are familiar and for which our tools, or thought processes, are familiar.

The very existence of the "distributed computing" model is a reaction to the last big wave, the one where the impact of these sorts of curves moved the sweet spot in computing-for-competitive-advantage from the mainframe data center to the desktop. UNIX happened to be, by a combination of good luck and rare foresight, in exactly the right place at exactly the right time, and so UNIX was a central player in that "revolution." The pendulum is swinging again, and it is the three curves, or more precisely, the economic implication of the three curves that is powering the swing.

Last August I heard Jack Valenti, head of the Motion Picture Association of America, boast that he could re-ignite the economy with only two ingredients, two ingredients he was challenging his audience to provide: 20 million homes on broadband and a reasonably viable solution to digital piracy. With that he could open a fixed-subscription-price video-on-demand service and use up the glut of bandwidth now in the ground. He might be right about what those two things would give him and about the market opportunity; Excite reports that already 15+% of its total bandwidth consumption is due to Kazaa. Boastfulness aside, Valenti is looking at near-term viability of a model where it is the bandwidth that is cheaper than the storage, which is cheaper than the processing power; prices are falling with a rapidity that only a couple of years ago all but a few would have found laughable. This is the three curves in action, as interpreted by an entrepreneurial schemer of the first rank.

As a different example, peer-to-peer is just beginning, and it is these curves that will drive it. Speaking to the system administrators in my audience: what do you think it will mean to have little computationally smart network nodes popping up everywhere, talking to each other, thinking nothing of exchanging vast quantities of data that appear to have zero cost to everyone except the infrastructure's cost of reliability? Security people: what do you think a network with no perimeter and impossible synchronization means to the data integrity and confidentiality constraints that more or less are the constants in every problem statement you get to solve? Database administrators: IBM Research expects 85% of all storage at IBM to be on SANs within five years – do you see the same thing coming and, if so, what are you going to do about organization much less naming?

I don't think we can be content to get better and better at what we do. I think we have to work ourselves out of a job in the sense that the essentialness of people like us just is not scalable. You've probably seen the study that suggests, given current growth rates (those three curves) in volume and complexity of computing, that half the world's population will have to be in computer operations, broadly defined, within the next 10 years. Similar predictions several decades ago removed telephone operators from the scaling limits of the telephone system through the introduction of direct dial. Before that, 100,000 elevator operators were made redundant by the automatic elevator, and it's a good thing as we now have four orders of magnitude more elevators today than we did then. We have to work ourselves out of a job or face becoming, ourselves, a limit to productivity growth.

It's a sobering thought. If not us, who?

Fifteen Years Ago in ;login:

by Peter H. Salus
USENIX Historian
peter@matrix.net

Nowadays, OSes seem confined to the UNIX model (in which I subsume Linux) and the VMS model (which includes Windows). But once upon a time in a universe alien to this one . . .

The January/February 1987 issue of *;login:* carried, among other things, an article by Matt Bishop on "How to Write a Setuid Program" and an article on the "Sprite Project," by John Ousterhout, A. Cherenson, Fred Douglis, M. Nelson, and Brent Welch.

The March/April issue contained articles on MINIX, by Andy Tanenbaum, and DASH, by Dave Anderson and Domenico Ferrari.

For those of you who have been living in the DOS/Windows/NT universe, Sprite was a network-oriented operating system, MINIX was (is?) a UNIX clone (compatible with V7) that would run on the IBM PC. The internal structure was quite different from UNIX in that it "is a message-passing system on top of which are memory and file servers." DASH was a very large distributed system, potentially involving "thousands or millions of connected hosts . . . [which] may be thousands of miles apart."

I consider these truly remarkable landmarks. Without MINIX we would (most likely) not have Linux, and Beowulf, SETI, etc. owe their bases to Sprite and DASH.

But these weren't all. In January/February, Marc Donner reviewed *The C Programmer's Handbook* and in

March/April, Lou Katz reviewed the very first "nutshell handbooks" from O'Reilly.

And there was the Weirdnix competition, too.

Quirky Cows & Computing Challenges: The USA Computing Olympiad

by Gary and Steven Sivek

[*Identical twins Gary Sivek and Steven Sivek have been participating in the USA Computing Olympiad for three years. Both have qualified twice for the summer training camp; last year, as a member of the US team that competed at the International Olympiad in Informatics, Steven earned a bronze medal.*

In this article, Gary and Steven explain why this contest keeps them coming back and why other teen programmers should join them. Ed.]

Bessie the Cow wants to use a pogo stick to travel along a cow path of integer length L. Bessie starts at point 0 and proceeds in integer pogo-jumps to land exactly on point L. Bessie's velocity, V, starts out at zero and is always nonnegative. At the beginning of each bounce, she can change her velocity by -1, 0, or +1. The velocity is the distance Bessie travels along the path during the bounce. Bessie can be traveling at any nonnegative velocity when she lands on point L. Bessie wishes to avoid jumping on any of the cow pies that happen to be located at various integer points along the path (not including location 1 or location L, of course). Your job is to determine the smallest number of bounces to reach exactly the end of the path, point L.

Thus begins a problem entitled "Cows on Pogo Sticks" from the USA Computing Olympiad's 2000 Fall Open. Accompanied by three other programming tasks, this formed part of a five-hour contest designed to challenge the best high school programmers from around the world. This was typical of the contests conducted since USACO's inception in 1993: problems differing in difficulty and solution methods, a wide range of scores among participants, and, of course, bizarre bovine humor.

These USACO contests simply wouldn't be the same without cows. These animals, abundant in the olympiad's home state of Wisconsin, the site of its annual training camp, are featured in nearly every problem as they engage anthropomorphically in a number of activities: attending "kinergarten," planning an L-shaped pool for their forest, and jogging around their pasture, among other things. Sometimes the benevolent Farmer John plays a pivotal role in the problems: trying to give cows gifts according to their wish lists, dividing the herd into a set of fields, or just calling the cows home for dinner. If you understood the "kinergarten" pun – kine is an archaic plural of cow – you'll love the bovine touch that helps USACO stand out among the other science olympiads. These contests have another purpose beyond simple Holstein hijinks and Guernsey glorification, of course: each year, they ultimately determine the top

l. to r.: Steven Sivek, Gary Sivek

15 programmers (using C, C++, or Pascal) in the United States, who win a trip to the University of Wisconsin-Parkside as finalists. There, after a week of intense

training and contests, the four-person team is selected that will represent the US in the annual International Olympiad in Informatics (IOI). The US team has historically excelled in competition. Last year's team of Reid Barton, 18; Tom Widland, 18; Vladimir Novakovski, 16; and Steven Sivek, 17, won the top gold medal, two silver medals, and a bronze medal, respectively.

The 2001 training camp was a nonstop flurry of activity for the finalists, selected after an unusually grueling US Open in which only one competitor received even half of the points possible. After arrival and introductions on the first day, every day began with either one of four three-hour experimental contests or one of two exhausting five-hour "Challenge Rounds" used to determine the team of four. Afternoons included discussions of solutions from the morning's contest, planning for the next day's contest strategy, lectures in different techniques, and, of course, time for relaxation activities such as ultimate Frisbee, disc golf, an annual business simulation game, and various excursions to movies, museums, and water parks. Nights included presentations by the coaches on hot research topics in computer science such as operating systems, network security, and mapping the Internet, which all showed interesting applications of computer science outside of these contests.

The 2001-2002 season has had a strong start, drawing almost 500 competitors in recent contests. The contests are scored by a unique grading algorithm that spreads scores over the full range of 0 to 1,000 possible points by carefully weighting programs and individual test cases according to relative difficulty; thus, a score of "only" 500 would actually indicate a strong performance, and a perfect 1,000 is exceptionally difficult to obtain.

Two divisions allow for a wider range of skill levels, with an orange division for those just beginning and a green division for more experienced students. Only green division participants are considered for the training camp, but students can switch to green mid-year and even earn an invitation from a strong performance in the U.S. Open alone! The U.S. Open is the final contest of the year, after the Fall, Winter, February, and Spring contests, and is administered by proctors in schools over a period of five hours.

Of course, it's easier to be interested in these contests than it is to excel in them. If you were confused by the above problem, you're not alone. But there is something you can do about that! Ever since the Winter 2000 contest, the coaches and some top-performing students have worked to provide an interactive training site online (*http://ace.delos.com/usacogate*) where you can try your hand at these problems and learn techniques to solve them. The page provides resources on such topics as the ever-popular dynamic programming (affectionately called "DP"), shortest path algorithms, greedy algorithms, network flow, effective search techniques, minimal spanning trees, computational geometry, and any other problem type that contestants might encounter, including the mysterious "ad hoc" category. It also makes students work completely through even the toughest challenges, not allowing anyone to move forward without completing preceding problems, but coaches are available for hints when needed. The solution to "Cows on Pogo Sticks" . . . well, we won't spoil the fun for you. Learn about DP in the training pages and solve it for yourself!

The USA Computing Olympiad would not be possible without the hard work of many people, including director Don Piele; head coach Rob Kolstad; coach Greg Galperin; coaches/former participants Hal Burch, Russ Cox, and Brian Dean; and USACO's generous sponsor, USENIX, which provides funding for the Olympiad each year. Beyond them,

though, USACO would not exist without its participants. So visit the USACO Web site for more information, and start competing!

USACO Contests

The USACO offers multiple Internet contests in which individual precollege students have three to five hours to solve three to five problems. Students are encouraged to, but do not have to, participate in all of these contests before entering the US Open. The US Open will be given on April 11, 2002, at local high schools.

Based on their performance in any contests they complete as well as their work on training materials, 15 students are selected for the USACO training camp. Four students from the training camp will form the US team that will travel to the International Olympiad in Informatics in August, to be held this year in Korea.

Visit *http://www.usaco.org* for more information, including past problems and instructions for joining the mailing list.

Research Exchange Program (ReX) Update from the Field

by Sabine Buchholz

buchholz@kub.nl

A report on the ReX exchange program between Tilburg University, the Netherlands, and the Natural Language and Information Processing (NLIP) Group at the Computer Laboratory, University of Cambridge, UK.

Since not all readers of *;login:* might be familiar with the research field of computational linguistics, which forms the scientific background of this report about my four-month exchange stay at Cambridge University, I will start by introducing some of the most important concepts. Computational linguistics studies the combination of computers and natural (i.e., human) languages. It aims at developing and implementing models of how natural languages can be processed. Applications include text-to-speech, machine translation, question answering, and natural-language interfaces. A common subtask in many applications is parsing: determining the syntactic structure of a sentence. Although to a certain extent parsing can be done on the basis of knowledge about the based on part-of-speech (like verb, noun, preposition) of words, it is widely acknowledged that information about specific words (lexical knowledge) is advantageous. One of the most important pieces of lexical information is subcategorization(subcat), especially of verbs. This tells us, for example, that a verb like "give" preferably takes two complements (the di-transitive frame): "to give somebody something"; whereas "invent" takes one complement (transitive): "to invent something"; and "sleep" takes none (intransitive): "to sleep" but not "to sleep something." This information helps the parser in disambiguating sentences that would otherwise be ambiguous, like "She gave/invented Tim water." As parsers should be applicable to all kinds of texts, from all domains (for example for applications on the Internet), and extensive subcategorization information is not readily available for all verbs, it can best be acquired automatically. It is this subcat acquisition problem that I worked on during my time at Cambridge.

Ted Briscoe is a reader in computational linguistics in the Natural Language and Information Processing (NLIP) Group at Cambridge University. He had previously developed an automatic subcat acquisition system that works by parsing large amounts of texts (parsing based on part-of-speech information only), recording the frequency with which each frame occurs with each verb, and filtering out combinations that did not occur sufficiently frequently (and thus probably result from parser errors). Those verb-frame combinations that pass the filter, together with their associated frequencies (converted to probabilities), can subsequently be used for better probabilistic parsing.

To improve the performance of this last filtering step, PhD student Anna Korhonen developed a method for smoothing the acquired frequency distributions of new verbs by backing-off to semantically related known verbs, and for filtering based on the maximum likelihood estimation (MLE) of the resulting frequencies. As her method presupposes knowledge about semantic classes of verbs that is not easily available for all verbs, my task for the project was to explore alternative filtering approaches using machine learning.

I was a fourth-year PhD student at Tilburg University, the Netherlands. Since pPart of my research concerns concerned finding grammatical relations between verbs and their complements, which is related to parsing and subcat acquisition, . I thus already knew several of Ted's and Anna's publications on the subject when Ted asked my supervisor Walter Daelemans whether one of Walter's students would be interested in the project. Walter had developed a machine-learning algorithm called Memory-Based Learning (based on the k-Nearest Neighbor algorithm) which I had also used for my thesis research, so I had the necessary background for the project and also liked the idea of spending some time in a foreign country. On the one hand, Cambridge with its beau-

tiful and famous university, dating back to the 13th century, is very different from the "modern industrial city" Tilburg with its 75-year-old university. On the other hand, everybody cycles there tooin Cambridge, and the landscape is conveniently flat and the city small, so I immediately felt at home. I arrived in August, which is a good time for getting to know the city, the river, and the surroundings but a bad time for arriving at a university since half the staff is on holiday or attending conferences or summer schools. My first weeks were complicated by the fact that the entire computer laboratory, of which the NLIP Group is a part, was moving to a new building at the western edge of the city (an event which should have happened long before I arrived but which had been postponed several times). So I started by (re)reading the available literature, most notably Anna's nearly finished thesis, and by discussing a lot with Ted and Anna. Once I got my own office and computer in the new building, I started to locate all of the corpus resources, acquisition system modules, and evaluation software I had been reading about, and to use them myself. I also had a look at the source code, reviving my knowledge of Lisp, C, and shell scripting on the way. I then worked on three the following topics:

After the subcat acquisition system has parsed a text, tokens of frames together with verbs can be extracted. These must then be classified into types of frames. For example, "He invented the telephone" and "The telephone was invented" instantiated the same kind of (transitive) frame. I adapted the classifier to accomplish this passive-to-active conversion, so that all the tokens would contribute to the frequency count of their mutual type.

To evaluate how well the subcat acquisition performs, a so-called gold standard had been created manually. This means that some verbs were chosen at random, people looked at a representative number (mostly 300) of sentences in which these verbs occur, and noted how often they occur with which frames. Performance of the system is then computed in terms of precision (how many of the verb-frame pairs that the system proposes are also in the gold standard), recall (how many of the pairs in the gold standard are found by the system), and rank correlation (how similar is the order of pairs if gold standard and system results are ordered by frequency).

However, there are more types of frames that should be distinguished on theoretical grounds than the subcat acquisition system is able to do on the basis of part-of-speech information alone. Therefore the output of the system frequently was not a list of frames for each verb but a list of frame disjunctions.

These disjunctions complicated the computation of precision, recall, and rank correlation. In addition, they made the results of evaluation of the machine-learning experiments hard to judge, since the learner tended to predict all possible disjunctions containing common frames. I therefore developed a variant of the classifier that in which it wasis forced to return a list of single frames (no disjunctions). It is an open question what would be the best way to make such a forced decision. At the moment, the most general or frequent frame of a disjunction is chosen.

I used a supervised machine-learning algorithm. This means that one part of the gold-standard material was used for training the algorithm and another part for testing it. For each verb-frame pair acquired by the subcat acquisition system, the learner has to make a binary decision: keep it or reject it. As a first step, I had to create a machine-learning instance for each such pair. Features of the instances correspond to pieces of information from system output or from external information sources (like the semantic classes used by Anna). I could then study the influence of various (combinations of) features on filter performance. After machine-learning filtering, instances need to be converted back to the initial format of lexical entries. Results are that the influence of features depends heavily on the sort of verbs tested. In general, a combination of type of frame and observed frequency performs well, and adding additional information about semantic classes helps a little. For a special group of verbs, however, the type of frame feature alone is sufficient and adding frequency information degrades performance. An experiment that still needs to be performed is to combine Anna's back-off smoothing with the machine-learning filtering.

I devoted much of the last part of my exchange into documenting my software so that research into the method can be continued after the official end of the exchange project. The documentation will form part of a larger technical report that should describe the subcat acquisition system and related modules. I will also use my new knowledge to make comparisons between the subcat acquisition system and parts of my thesis work.

I had a very pleasant and informative stay and wish to thank all the people who made my exchange visit possible.

For more information about this exchange, please contact:

Dr. Sabine Buchholz
S.Buchholz@kub.nl
Dr. Ted Briscoe
Ted.Briscoe@cl.cam.ac.uk

USENIX and Stichting NLnet jointly support the ReX program. For more information about ReX, see *http://www.usenix.org/about/rex.html*.