Paper 279-26

GROWING GENETIC COWS WITH SAS: USING LOW COST PARALLEL PROCESSING WITH LINUX – MS/NT NETWORKS TO SEARCH GENOMES

Haftan Eckholdt, DayTrends, Brooklyn, NY

ABSTRACT

Using SAS to manage and run a LINUX / WINDOWS-NT based cluster of workstations (COW) to simulate a parallel processor puts the price of supercomputing well within most research budgets. The criteria for using COW's will be discussed in terms of their job symmetry and intermediate solution timing. Special problems, and their solutions are discussed including the fact that all COWs need: lots of smoothed electricity, lots of cool breezes; and lots of hard drives. Using three layers of file monitoring, job distribution, and data analysis, SAS can provide clever and cost effective paths to COW management and usage [X command, %MACRO, DATA, STAT].

Parsing the problem, and distributing it symmetrically across the COW and executing the solution are discussed in the context of an example from genetics: a search for a family a proteins in the entire genetic sequence of a worm, Caenorhabditis elegans, initially referenced from the fly (Drosophila melanogaste). Using a modified distance profile algorithm to assess amino acid periodicities, 12,000 proteins were characterized and ranked to yield 23 familial candidates. Readers can goto http://www.thismind.com/haftan to see drafts of a more detailed description of these findings.

INTRODUCTION

Research in disciplines with high density data have been limited by the state of hardware and software in supercomputing. Examples include simulations, model building, and graphics in weather, finance, and genetics. The barriers to supercomputing have traditionally been costs associated with hardware and programming. Using a cluster of workstations (COW) to simulate a parallel processor puts the price of a teraflop of hardware under $1,000,000. More importantly, perhaps, is that the scalability of a COW -- 50 gigaflops for $50,000 -- well within most research budgets.

The growing history on parallel processing can best be accessed through the Computer Society of Institute of Electrical and Electronics Engineers (IEEE) web site [http://www.computer.org/parascope/] which also sponsors the International Symposium on Parallel Architectures, Algorithms and Networks. The state of the art can be summarized by the simple return of 66,500 URL's from the www.google.com search for "cluster of workstations" as of September 2000. Readers should refer to the brief bibliography for recent text books on the topic.

The real criteria for using COW's are the nature and eligibility of the problem under study. Approaches to parallel processing, in bioinformatics, are classified by the instruction set (single versus multiple) and data set (single versus multiple) to yield four types of parallel processors SISD/MISD/SIMD/MIMD (Buyya, 1999b; Foster, 1994; Wilkinson and Allen, 1999). The distinction between these four types of processing are not always obvious, so it seems that using SAS to manage and run the COW gives the user access to all four approaches, depending upon which one is logically possible, and then, which one is most efficient. In general, questions that lend themselves to parallel processing on a COW are those that can be broken down into parts or jobs. "Granularity" is what COW programmers use to describe just how small, how similar, and how many jobs can be parallel-*ized* from the overall problem. It does make coding much easier if jobs do not depend on eachother.

Jobs that are the same in terms of analysis would require the same instruction set (single instruction) copied to each workstation, while jobs that are analytically different require different instruction sets (multiple instruction). Likewise, the data demands of the jobs will determine whether the same data set is used (single data) by all workstations or different data sets are used (multiple data).

Most importantly, there must be at least one job for each processor / workstation, no matter how large or small the individual job may be. Those problems that benefit most from parallel processing COWS are those that have either: (1) a large number of symmetric jobs, or (2) jobs that benefit from eachothers' intermediate solutions. Problems that fit criteria (1) are problems that can be distributed rapidly and evenly across the COW, whereby no processor is idle, and each job solution is posted back to the server. The type (1) problems are those that have what might be considered a large set of non-dimensional jobs. This long list of jobs must be completed for the problem to be solved, but there is no logical or necessary order in which the jobs must be completed. In this example, a COW outruns a SoW (single workstation) in that the throughputs (area/access to critical devices: Hard Drive, RAM, CPU) have been radically expanded, and repeated in parallel. In some cases, the resulting throughput of a COW exceeds that of a similar capacity supercomputer. Type (2) problems are problems that benefit from the "folding of time" that can occur in a parallel process. For instance, some solutions might occur faster with access to solutions from jobs later in the process, something that would never occur under traditional computing environments – SoW/supercomputer.

This talk, like most books on the topic, will fall back on demonstration to explain the challenge of problem parsing. The following sections will provide a more detailed description of the process of setting up and using a COW including hardware designs, and cluster management, and job execution. Readers should note that the overall approach to cluster management and job execution is a patent pending technology, although this paper provides specific approaches to cluster management and job execution using SAS.

HARDWARE

There are several essential layers constructing a COW. A fileserver is needed to hold code, hold/distribute data, collect answers. A primary workstation is needed to write code, submit code (direct/ftp/etc), collect answers, proxy server in peer to peer network. And finally, many workstations are needed to share executions in parallel cluster of workstations. Other issues to be discussed include sharing peripherals (monitors, keyboards, and mice), power, cooling, and communications. Readers should note that the approach to parallel COW processing discussed here is not very efficient in terms of network communications. Making clusters more efficient is discussed in great detail in Buyya (1999).

FILESERVER

First of all, the fastest and cheapest file server is LINUX! Not only is Linux inexpensive, but it is easy to set up. And, most importantly, there is plenty of research to show that Linux servers are very fast. Once the number of clients exceeds about 8 workstations, there is no competition.

In this case, Red hat 6.2 [www.redhat.com] was downloaded and installed on a modified station (faster throughput with two hard drives [primary master + secondary master] set in RAID0 configuration with two Ethernet cards). You can purchase RedHat 7.0 with a beautiful new user interface, or you can download the very same thing for nothing. RedHat Linux has no limits on the number of attached clients. Buying support is (a) not very necessary, and (b) not very useful. This Samba Server, necessary to connect Microsoft NT workstations, was up and running in 12 minutes. Readers should note that I have faked this part of the COW in the past with a plug and play SNAP! Server which has no limits on the number of clients as well. For long standing COWS, the higher speed and life of a traditional fileserver is worth the additional cost.

WORKSTATION

Any Number of NT workstations were linked in a traditional client/server architecture. The configuration of each workstation was identical. Actually only one drive was set up and optimized, and the rest were cloned across the LAN with Symantec GHOST in 45 minutes! GHOST is part of *NORTON SYSTEM* WORKS, and a shareware version can get you most of the way there for nothing, although the full function version is worth the price. Note that each workstation had a market value of about $450 in September 2000 (500 Mhz, 128 Mb RAM, 10 Gb HDD, 100 mps Nic, SVGA).

COMMUNICATIONS

Workstations shared several keyboards/mice/monitors with Belkin KVM switches. Remember that none of workstations or file server will be touched again after the first login in most cases. The network communication was handled as cheaply as possible using small hubs ($100 each). Long standing COWs should use more expensive buy faster network switches ($1000 each).

POWER & ENVIRONMENT

A few cautions should be noted: (1) all COWs need lots of smoothed electricity (many smaller UPS's are significantly cheaper and a single larger UPS); (2) all COWs produce lots of heat, so there needs to be lots of cool breezes; and (3) in the event of a power outage, the UPS should be set to shut down all systems before the ambient temperature reaches critical limits… a few minutes in most rooms is a good rule of thumb, really; (4) all COWs eat hard drives for lunch, so use swap drawers, have plenty of GHOSTed clones on hand, and carefully monitor the warranty on any drive [hint: an IDE drive on a COW will definitely burn up before its warranty expires].

SOFTWARE

Overcoming the software barrier is the last hurdle, and SAS can provide and clever and cost effective paths to COW management and usage. Readers should note that lots of other programs could be used to manage and execute jobs on a COW (Perl, C++, etc.) and that there are companies selling cluster management software. Managing a COW, with SAS, is the point of this story.
In this example, each workstation needs a copy of SAS. SAS will be used to manage the cluster, submit jobs, distribute jobs, share data, and collect solutions. The general approach in the COW workstation flows like this:

1. Learn identity, location in network, and size of network.
2. Look for a new job to execute from server.
3. Get necessary data from server.
4. Execute relevant section of job.
5. Post solution to server.
6. GOTO step 2.

STARTUP

SAS was set in the NT startup folder so that SAS initiates after login. A startup file placed in autoexec.sas should minimally contain the unique identity of the workstation, its place in the network, and %MACRO loop to check a reference file, and barring changes, go to sleep for a while.

```
* future X commands should not wait;
OPTIONS NOXWAIT;

* log files will stop a cow in it tracks;
FILENAME NOLOG DUMMY;
PROC PRINTTO LOG=NOLOG;

* establish unique machine identity from
local directory name;

%MACRO MID;
* send directory name to text file;
X "DIR D:\SASSPACE > :\SASSPACE\MACHINE.TXT";

* acquire text file :: work in MSNT4;
DATA WORK;
INFILE "D:\SASSPACE\MACHINE.TXT" FIRSTOBS =
8;
 INPUT CREATED MMDDYY8. CREATET TIME7. AMPM $
FORM $ MACHINE $;

IF FORM = "<DIR>";
IF UPCASE(SUBSTR(MACHINE,1,2)) = "DT";

DATA WORK;
 SET WORK;

* parse the identity;

NUM = 1*SUBSTR(MACHINE,3,3);
TEN = 1*SUBSTR(MACHINE,4,1);
ONE = 1*SUBSTR(MACHINE,5,1);

IF NUM = . THEN DELETE;

DATA WORK;
 SET WORK;

CALL SYMPUT("HOME",TRIM(MACHINE));

PROC PRINT;
PROC PRINT;

%MEND MID;
 %MID;
```

After the identity is learned, each workstation can begin monitoring the server for a new SAS command file containing new work. This is accomplished by comparing the file creation information repeatedly: (1) going to sleep if there are no changes, or (2) copying and executing the file if it changes.

```
* get the initial estimates of file creation
date and time;

X "DIR X:\DNA\ > D:\SASSPACE\TIME2.TXT";

DATA TIME2;
 INFILE "D:\SASSPACE\TIME2.TXT" FIRSTOBS = 8
LRECL=65 PAD;
 INPUT CREATED $ 1-8 CREATET $ 11-16 FILEN $
40-49;

FILEN = UPCASE(FILEN);
IF FILEN = "DOTHIS.SAS";

FTIME = CREATET;
FDATE = CREATED;
TIME = 2;

 DATA _NULL_;
```

```
  slept=sleep((60*60*0)+(60*&TWAIT));
 RUN;

* begin looping procedure to assess changes
in file creation date and time;

%MACRO WORK;

 %DO I = 1 %TO &TSTOP;

X "DIR X:\DNA\ > D:\SASSPACE\TIME1.TXT";

DATA TIME1;
 INFILE "D:\SASSPACE\TIME1.TXT" FIRSTOBS = 8
LRECL=65 PAD;
 INPUT CREATED $ 1-8 CREATET $ 11-16 FILEN $
40-49;

FILEN = UPCASE(FILEN);
IF FILEN = "DOTHIS.SAS";

FTIME = CREATET;
FDATE = CREATED;
TIME = 1;

DATA TIME12;
 SET TIME1 TIME2;

FTIME1 = LAG1(FTIME);
FTIME2 = FTIME;
FDATE1 = LAG1(FDATE);
FDATE2 = FDATE;

DATA TIME12;
 SET TIME12;

IF TIME = 2;
IF FTIME2 NE FTIME1 OR FDATE2 NE FDATE1 THEN
DIRX = "DNA";
IF FTIME2 = FTIME1 AND FDATE2 = FDATE1 THEN
DIRX = "DNX";
 CALL SYMPUT ("DX",DIRX);
  ROUND = &I;

DATA TIME12;
 SET TIME12;

%INCLUDE "X:\&DX\DOTHIS.SAS";

 DATA _NULL_;
  slept=sleep((60*60*0)+(60*&TWAIT));
 RUN;
```

EXECUTION

The existence of the reference file is critical to SAS/COW management. This reference file (here called DOTHIS.SAS), which does not contain the actual job, contains pointers to the real COW jobs. Using the reference file allows the user to overcome network level file access violations, and allows the user to point to any number of new or different COW job on the network without touching any workstation. The reference file in this example utilize operating system shell commands to calculate the number of jobs, and to symmetrically distribute the jobs across the COW using commands like following:

```
***> SET JOB VARIABLES;

%LET CPU = 100;
%LET CODE = MODELA;
%LET SOURCE = WORMS;
LIBNAME DNA "X:\DNA\";
LIBNAME HOME "X:\&HOME\";
LIBNAME TEMP "D:\SASSPACE\";

%MACRO JOBS;

X "DIR X:\DNA\%SCAN(&SOURCE,1)\*.TXT /O:N >
D:\SASSPACE\SOURCE.TXT";

DATA TEMP.SOURCE;
 SET DNA.MEMBER;

* INFILE "D:\SASSPACE\SOURCE.TXT" FIRSTOBS =
6  LRECL=65 PAD;
* INPUT CREATED $ 1-8 CREATET $ 11-16 MIDDLE
$ 17-39 FILEN $ 40-49;

IF COMPBL(MIDDLE) = "<DIR>" THEN DELETE;
IF CREATED = " " THEN DELETE;

NAME =
COMPRESS(SUBSTR(FILEN,1,INDEX(FILEN,".")),"."
);

DATA _NULL_;
 CALL SYMPUT ("JOB",COUNT);
 STOP;
 SET TEMP.SOURCE NOBS=COUNT;

run;

%MEND JOBS;
 %JOBS;

* CUT UP AND ASSIGN THE WORK LOAD;
%MACRO WORK2;

DATA WORK;
 SET WORK;

PART = INT((&JOB/&CPU)+1);
START = (NUM-11)*PART + 1;
STOP  = START + PART;

DATA WORK;
 SET WORK;

CALL SYMPUT("MSTART",START);
CALL SYMPUT("MSTOP",STOP);

%INCLUDE "X:\DNA\%SCAN(&CODE,1).SAS";
```

DEMONSTRATION

Of course, the kewlest part of this story is an example of using a SAS managed COW to search for a family a proteins in the entire genetic sequence of a worm, Caenorhabditis elegans. This section will be the focus of a presentation at another conference.

PROTEINOMICS

Few people on the planet are truly intimate with scope and nature of the tasks set before community of gene scientists. Most genetic scientists working in a traditional laboratory setting deal in actual gene sequences of 100 to 1,000 units in length at a very physical level. Bioinformatic research today is primarily dedicated to computerized tools designed to aid lab geneticists in an almost post hoc investigation of well characterized regions, genes and species. Rare is the bioinformatic researcher that sets out to investigate almost exclusively in computer / virtual laboratories. These a priori investigations are accounted in terabytes (1 billion kilobytes of computer disk storage) and gigaflops (1 billion operations per second on a computer's central processing unit). Such researchers discuss petabytes and teraflops in casual conversation without appreciating the implication of these unit shifts on our understanding of genetic information. Genetic scientists using genes, algorithms, and computers are at the very bottom of very steep learning curve. Today we are not just unprepared but also unfamiliar with what we will be doing tomorrow.

Computer based analysis of genetic data is used for two distinct problems: finding proteins (searching / matching sequences) and building proteins (determining shape and structure). These distinct problems come together when determining protein function. Understanding protein function is of course, the basis for treating disease. This application is primarily concerned with protein searching and matching, although the transfer of technology described below to assessing structure and function is possible.

Available matching engines provide users with powerful tools for finding genes based upon targeted sequences, be they WEB based or licensed for standalone workstation. Each search engine suffers from several common problems in (1) sequence dependency, (2) estimation bias, and (3) computation intensity.

DISTANCE PROFILING

Using a modified distance profile algorithm to assess amino acid periodicities (initially described in a chapter by Konopka on biomolecular cryptology in Smith, 1994), in the six chromosomes of C elegans (downloaded from the National Library of Medicine [ftp://ncbi.nlm.nih.gov/genbank/genomes/C_elegans/].

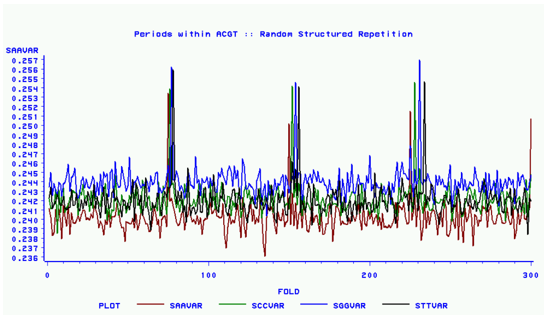
For the purposes of this application, an example of traditional gap analysis in genetics can be drawn from simple comparisons within or between species. Below is matrix of text that demonstrates the design and scoring in a simple gap function whereby the target string on top is scored by various criteria identified on the left.

Table 1. Example of traditional scoring in gap analysis.

Sequence	Gap	A	C	G	G	A	A	A	C	T	A	A	C
AA	1	0	0	0	0	0	1	1	0	0	0	1	0
AA	2	0	0	0	0	0	0	1	0	0	0	0	0
AC	1	0	1	0	0	0	0	0	1	0	0	0	1
AC	2	0	0	0	0	0	0	0	1	0	0	0	1

The first criteria is the occurrence of AA. Any occurrence of AA is scored as 1. The next criteria called AA2 is actually searched as A*A whereby a 1 is scored whenever an A occurs two gaps down from an A. This scoring scheme can be carried out for any number of combinations, depending upon the nature of the question investigated. Traditionally, gap analysis is performed to many gaps, starting at the first (next door neighbor) and ending at some relevant neighbor (30, 300, etc.). Each string of 1's and 0's is then summed and the standard deviations or z-scores are graphed. The figure below was generated using a random string of 100,000 characters (ACTG) in length, with a "twin neighbor" (AA, CC, TT, GG) systematically placed every 75 gaps away.

Figure 1 . Analysis of twin event in 300 gaps of random/artificial string.



MODIFIED DISTANCE PROFILING

In the proposed solution, I have modified traditional gap analysis whereby the specific sequence is ignored, but rather the general characteristic of self/same or twinning is scored. Directly below is the same matrix presented earlier, but with the new scoring algorithm. When this scoring algorithm is used, much of the diagnostic and comparative power is retained while the analytic novelty is dropped.

Table 2. Example of modified scoring of gaps emphasizing "twin events".

Sequence	Gap	A	C	G	G	A	A	A	C	T	A	A	C
Twin	1	0	0	0	1	0	1	1	0	0	0	1	0
Twin	2	0	0	0	0	0	0	1	0	0	0	0	0
Twin	3	0	0	0	0	0	0	0	0	0	1	0	0
Twin	4	0	0	0	0	1	0	0	0	0	1	1	1

The proposed approach to gene searching has less sequence dependency, relative to existing methods, because the sequence itself, although "entered" in the algorithm for calculations, is not used for the actual protein search. Rather, the statistics or metrics about the comparative frequency of twinning at each gap length are used. Furthermore, the metrics about twinning at each gap length are not character specific. Locally, the AA is given the same score and weight as the CC, the GG, and the TT as seen in the matrix above.

Another strength of sequential dynamics is that a protein of any length can be analyzed for comparison. In contrast to most existing engines that constrain the user to inputting relatively few bits of genetic material, this approach thrives on long sequence targets and indices.

Once characterized, each protein was then ranked by its spatial (not sequence!) fit to the reference protein (called SOP-3 by Zhang and Emmons).

PILOT DATA

An initial test of the utility this algorithm consisted of a search for "SOP like" proteins in c elegans using the established relationships between SOP-3 in c elegans (SOP-3) and SOP like proteins in drosophila (CNC, LAG3, MAM, and TSH). This search began with the download of all c elegans and relevant drosophila proteins from the WEB:

1. SOP-3 from the Emmons laboratory.

Sop-3, like Sop-1, plays a role in the neurogenesis of th email tail. It encodes a homolog of the mammalian Mediator complex protein TRAP230. In sop-1 mutants, pal-1 is activated by a pathway that is stimulated by bar-1/beta-catenin, a component of the Wnt signal transduction pathway. The results support a physiological role of the Mediator complex in conveying regulatory signals to the transcriptional apparatus (Zhang and Emmons, 1997; Zhang and Emmons, 2000).

2. Teashirt (tsh) of drosophila:

http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=Protein&list_uids=103315&dopt=GenPept

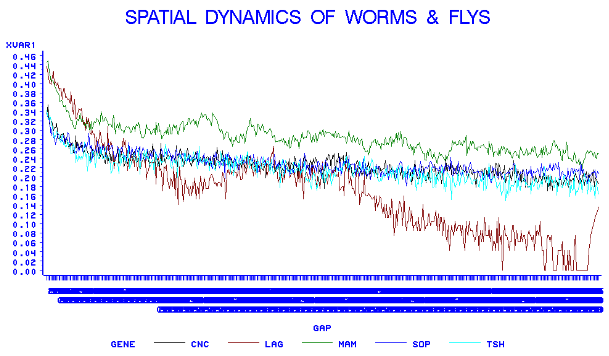
Homeotic genes control the development of embryonic structure by coordinating the activities of downstream 'target' genes. The identities and functions of target genes must be understood in order to learn how homeotic genes control morphogenesis. Drosophila midgut development is regulated by homeotic genes expressed in the visceral mesoderm, where two of their target genes have been identified. Mathies, Kerridge, and Scott (1994) reported the identification of another homeotic gene target in the midgut mesoderm, the teashirt (tsh) gene, which encodes a protein with zinc finger motifs. tsh is necessary for proper formation of anterior and central midgut structures. By responding to signals as well as localized transcription regulators, the tsh transcription factor is produced in a spatial pattern distinct from any of the homeotic genes.

3. Cap N Collar (cnc) of drosophila:
http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=Protein&list_uids=3859889&dopt=GenPept
The Drosophila cap 'n' collar locus controls mandibular and labral cells of the head during mid-stages of embryogenesis. Transformations are associated with persistent deformed expression in anterior mandibular cells. Cnc provides a mechanism to modulate the specificity of Hox morphogenetic outcomes, which results in an increase in the segmental diversity in the Drosophila head (McGinnis, Ragnhildstveit, Veraksa, and McGinnis, 1998).

4. Mastermind (mam) of drosophila:
http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=Protein&list_uids=103241&dopt=GenPept
The phenotypes and genetic interactions associated with mutations in the Drosophila mastermind (mam) gene have implicated it as a component of the Notch signaling pathway. However, its function and site of action within many tissues requiring Notch signaling have not been thoroughly investigated. Truncated Mam results in failure of lateral inhibition within proneural clusters and perturbations in cell fate specification within the sensory organ precursor cell lineage. Expression in the wing is associated with vein thickening and margin defects, including nicking and bristle loss (Helms, et al., 1999).

Using the “any twin” scoring of gap analysis illustrated above, we calculated the twin gaps from 1 to 300 of each protein of the 4 proteins listed above. MAM is at the top of the graphic while LAG is at the bottom. Almost obscuring eachother are sop, cnc, and tsh.

Figure 5. Twin gap analysis of five proteins from C. elegans and drosophila.



In order to find other “sop like” proteins in C. elegans, we calculated the twin gaps from 1 to 500 of each protein in the six chromosomes. This computationally intensive process was begun by downloading the sequences from ftp://ncbi.nlm.nih.gov/genbank/genomes/C_elegans/. For C. elegans, the following files and sizes are relevant to the computational tasks for all six chromosomes combined:

Table 3. Description of data needs for C. elegans twin gap engine library.

Size	Description
12 Mb	ASCII data structures from WWW site
12 Gb	temporary computation structures
3 Gb	intermediate structures retained for graphical events
2.5 Mb	final analytic structures used for gene searching

Each protein (12,000 total) was gap analyzed in approximately 120 seconds on a 500 Mhz CPU with 128 Mb of RAM using SAS version 6.12 running under Windows NT 4.0 workstation. Using a single workstation would have taken 16 days of continuous processing time. A 20 workstations COW would take 20 to do the same job.

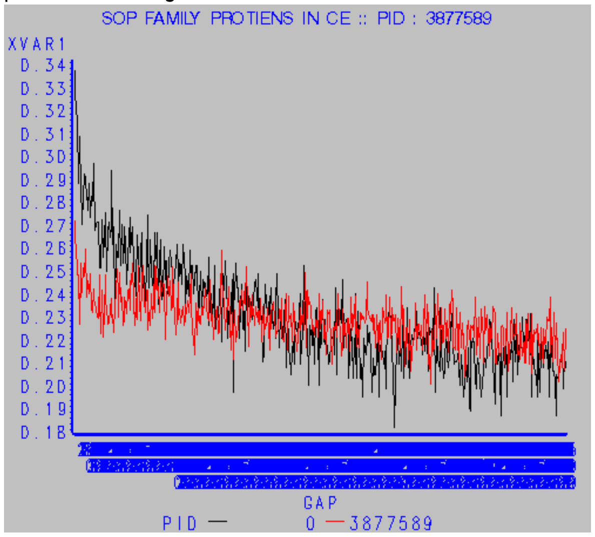
Once these metrics were calculated, the search for “sop like” proteins began in earnest. To find that candidates we utilized the “fit” between sop and the best fitting of the drosophila proteins, namely: cnc and tsh. This was accomplished by calculating a typical spatial event: the square root of the squared area between each gap metric (the sum of the distance between the two lines at each gap). This search was conducted on a single workstation and took 6 minutes of processing time. Here lies the essence of search engine, a process which will always take about the same amount of time (6 mintues on 1 workstation), a process that can also be ported to the parallel cluster. The resulting distances were sorted, and those proteins meeting or exceeding the “closeness of fit” between sop3 and tsh or sop3 and cnc were brought forward for human scrutiny.

Table 4. Results of twin gap search for proteins in C. elegans nearest to the sop/tsh/cnc fit.

CHROM	GENE	PID	LENGTH	START	DSUM
0	SOP		P		
0	TSH				
0	LAG				
0	CNC				
0	MAM				
1	T23H2.1	1703554	1848	4789480	-0.98419
1	C50F2.2	1707033	1047	2243036	0.52181
1	C44E4.1a	2088725	3865	2869201	-0.90799
1	F22G12.5	3876039	1566	13380478	-0.43828
1	lrp-1	3876533	4754	5958339	-0.21829
1	F36A2.13	3876547	1206	7180130	0.22220
1	F36A2.13	3876782	1709	7174161	0.22220
1	H05L14.2	3878009	2113	6327339	0.71326
1	Y47H9C.4	3881080	1112	12529845	0.29746
1	ZK1151.3	3881532	2020	11010414	0.00874
2	ZK430.1	1125755	1651	4358173	0.35704
3	F52C9.8b	1055055	1002	4111964	0.19919
3	ZK328.5b	1213538	1666	4826763	-0.54329
3	R10F2.1	2088852	2164	1817375	-0.12982
3	C38D4.3	3874824	1696	3577191	-0.88904
3	D2045.2	3875359	1793	9425358	-0.00710
3	F35G12.8	3876724	1550	3380816	-0.89717
5	C35C5.6	3874766	1225	11305541	0.61311
6	T23F2.2	1049372	1511	5231136	-0.99071
6	C54G7.3	1065455	2947	5259642	-0.38854
6	F07C7.1	1166600	1880	8961448	0.71446
6	F21E9.1	2315597	1171	1073742	-0.26735
6	F54E4.1	3877589	2949	14610013	0.87600

The first column designates the chromosome of origin on C. elegans (0 represents the index proteins). This return matrix can be accessed on the WEB at http://www.thismind.com/haftan.

Figure 6. Graphical output from twin gap search for "sop-like" proteins in *C. elegans*.



CONCLUSION

Characterizing and ranking approximately 12,000 proteins yielded 23 candidate proteins. Of these 23 candidates, one was identified by geneticists as being "of the family" that is defined by SOP-1 (the original), SOP-3, the newly isolated reference, as well as two known homologs from the fly (Drosophila melanogaste). Readers can goto http://www.thismind.com/haftan/ to see a detailed description of these findings. Note that these candidates could not be found using the available genome search engines on the WEB because they are sequence specific.

Using a Parallel Cluster of Workstations brings this approach to hypothesis testing from the concept (2-3 weeks) to reality (1 day). Science and industry are littered with questions that have yet to be asked because of computational burdens that no longer exist.

For certain kinds of problems, SAS has the computational power to conduct the necessary analyses, but more importantly, the operational complexity to manage the problem on a network of workstations. Combining problem solving and data, file, network, and problem management in one software platform provides a critical cost savings for many researchers.

REFERENCES

Buyya, R. (1999). High Performance Cluster Computing: Architecture and Systems, volume 1. Prentice Hall: New Jersey.

Buyya, R. (1999b). High Performance Cluster Computing: Programming and applications, volume 2. Prentice Hall: New Jersey.

Foster, I. (1994). Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison_Wesley Publishing Company: New York.

Harris, Z. (1954). Distributional structure. Word. 10: 775-793.

Helms, W, Lee, H, Ammerman, M, Parks, AL, Muskavitch, MA, Yedvobnick, B. (1999). Engineered truncations in the Drosophila mastermind protein disrupt Notch pathway function. Developmental Biology. 215(2): 358-374.

Hill, T. (1998). WINDOWS NT: Shell Scripting. Macmillian Technical Publishing: Indianapolis, IN.

Konopka, A. and Smythers, G. (1987). DISTAN – A program which detects significant distances between short oligonucleotides. Computers and Applied Bioscience. 3: 193-201.

Mathies, L., Kerridge, S. and Scott, M. (1994). Role of the teashirt gene in Drosophila midgut morphogenesis: secreted proteins mediate the action of homeotic genes. Development. 120: 2799-2809.

McGinnis, N., Ragnhildstveit, E., Veraksa, A. and McGinnis, W. (1998). A cap 'n' collar protein isoform contains a selective Hox repressor function. Devlopment. 125: 4553-4564.

Singh, S. (1999). The Code Book. Doubleday: New York.

Ulam, S. (1976). Adventures of a Mathematician. Charles Scribner's Sons: New York.

Wilkinson, B. and Allen, M. (1999). Parallel Programming: techniques and applications using networked workstations and parallel computers. Prentice Hall: New Jersey.

Zhang, H. and Emmons, S. (2000). A c. elegans mediator protein confers regulatory selectivity on linege-specific expression of a transcription factor gene. Genes Development. 14(17) 2161-2172.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Haftan Eckholdt
DayTrends
10 Jay Street
Brooklyn, New York 11201
Work Phone: (718) 522-3170
Fax: (718) 522-3170
Email: haftan@daytrends.com
Web: www.daytrends.com