Cruising the LeMans FunRun

Jason Secosky
Allan Snavely
Francine Berman*
Department of Computer Science and Engineering
University of California, San Diego

Tom Kitchens
Department of Energy

Abstract

The LeMans FunRun is a heterogeneous competition which was held at Supercomputing '93 in Portland Oregon. The participants raced to "solve" a puzzle in a fixed amount of time on a network of machines comprised of a CM-5, Paragon, MasPar, Cray and Silicon Graphics workstations. This paper describes the design and implementation of the LeMans FunRun, and discusses its potential for use as a heterogeneous educational tool. [1]

Keywords: Heterogeneous computing, network, parallel computing, educational software.

*This research was partially supported by the National Science Foundation contract number ASC-9301788.

[1] A revised version of this article was published in IEEE Parallel and Distributed Technology, Volume 2, Number 2.

1 Introduction

Heterogeneous Network Computing is an emerging area of computational science whose focus is the solution of complex parallel programs on networks of distinct types of processors. (We differentiate between *heterogeneous network computing* in which the processors are of different architectural types and *cluster computing* in which the processors are of the same architectural type.) The central thesis of heterogeneous network computing is that each machine performs some tasks better than others. By selecting the right machine for the right task, performance can be improved. Applications which can be partitioned as collections of tasks which exhibit distinct computational paradigms can potentially achieve greater speedup on a heterogeneous network of machines which support these paradigms than the entire application can achieve on any single machine in the network.

The LeMans FunRun was developed as a heterogeneous competition for Supercomputing 93 in Portland, Oregon. Five distinct machines on SCInet (an experimental network testbed on the exhibition floor of the conference) were used as a heterogeneous network to solve a "problem". The machines used were a Cray EL-98, Intel Paragon, Thinking Machines CM-5, MasPar MP-2 and Silicon Graphics Indy machines. The "problem" consisted of 9 tasks to be solved on specific machines in the network. The participant who solved the most tasks in a fixed amount of time (24 minutes, extended to 34 minutes during the competition) was declared the winner.

The LeMans FunRun provides a demonstration of heterogeneous computing and highlights its potential and problems. In this paper, we describe the design and implementation of the LeMans FunRun in Sections 2 and 3. In Section 4, we discuss the implications of the FunRun and its potential for development as an educational tool for heterogeneous computing.

2 The Competition

The goal of the LeMans FunRun is to accumulate as many pieces of a 9 piece ($3X3$) puzzle as possible in a fixed amount of time. Each piece of the puzzle corresponds to a task to be solved on a specific machine in the network. A participant receives a piece by correctly executing a task for a given input. Participants can obtain any of the 9 pieces as the first puzzle piece, however

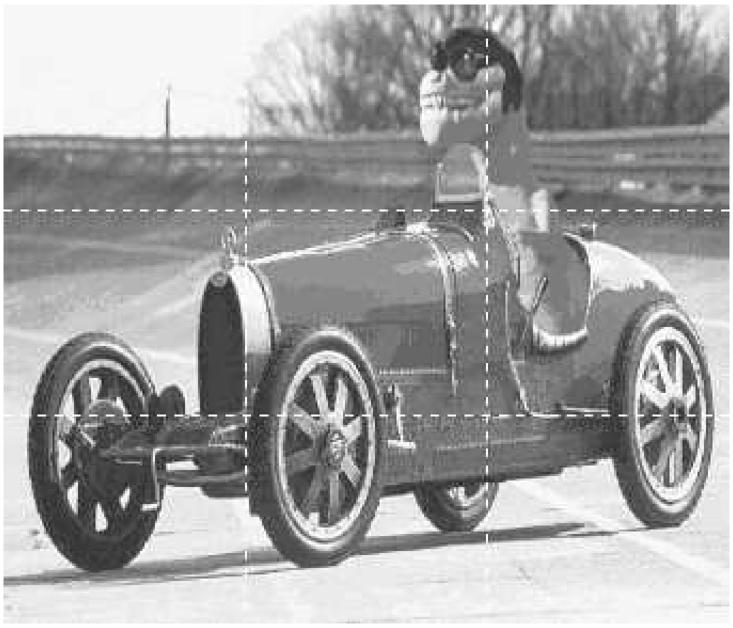

Figure 1: Puzzle solved during the LeMans FunRun at Supercomputing 93.

each subsequent piece accumulated must be adjacent to a puzzle piece which has already been received. The adjacency constraint places a dependence on the tasks. We did this by design because tasks within most heterogeneous applications are partially dependent. The puzzle "solved" by the participants is shown in Figure 1. The tasks which corresponded to each piece were non-uniform in size and execution time so a winning strategy would not necessarily start with the middle piece.

The FunRun is driven by the Flagman, a graphical user interface which supervises the competition, the network, and the participants. The Flagman is a central monitor or umpire program responsible for coordinating the participants' progress. The topology of the FunRun is that of a star network (Figure 3). The Flagman's machine is at the center of the star, with each computer in the network forming the points of the star.

The Flagman provides a graphical interface which displays the pieces currently accumulated for each participant's puzzle so that participants' progress may be compared. All puzzle pieces are timestamped as they are received and validated by the Flagman. (Incorrect output for the task executed is not considered valid and the participant must re-execute the task with new input on the target machine.) The timestamp of the last puzzle piece validated for each participant is recorded by the Flagman so that at the end of the competition, ties may be broken.

Participants were provided with descriptions of the problems to be solved and sample unoptimized codes for their solution three weeks in advance of the competition. The participants could use the sample solutions or produce code of their own to solve the problems. Participants were allowed to use any resources available to them in order to solve the problems. Any language could be used, however each problem had to be solved on the machine for which it was designated. During the competition, any participant found to be intentionally using machine resources for computation not related to solving a designated problem was subject to disqualification.

At Supercomputing 93, participants were given 24 hours prior to the competition to log in to the FunRun machines on the exhibition floor and compile their problems. Input for the first task chosen by each participant was provided at the start of the competition and inputs for subsequent tasks were provided during the competition after validation of a new puzzle piece. At the end of 24 minutes, participants had typically solved less than half of the tasks due to contention for machines or programmer error. The panel of judges decided to extend the competition 10 minutes. At the end of 34 minutes, two teams had eight pieces out of nine with a timestamp of 30 and 32 minutes, and two had six pieces with a timestamp of 28 minutes and 33 minutes. Freund and Campbell, whom led the team from Naval Research and Development (NRaD), won first place in the competition and were recognized at the Heterogeneous Computing Challenge Minisymposium and at the Supercomputing 93 Awards Luncheon.

3 Behind the Scenes: The FunRun Software

Designing and implementing the FunRun software involved the development of several distinct parts: the Flagman program, network and computer interfaces, and tasks to be solved on the various machines. In this section, we discuss the design and development of these parts in more detail.

3.1 Flagman

The Flagman is a program which runs on a workstation and gathers information from each of the machines in the network (Figure 2). It is the center of the star topology. The job of the Flagman is to decide the validity of solutions, provide additional inputs when problems have been solved correctly, provide alternative inputs and rudimentary error information when problems have been solved incorrectly, and to serve as a repository for a shared database of current competition information. The Flagman ran at Supercomputing 93 on a Silicon Graphics Indy workstation and was developed at UCSD and the San Diego Supercomputer Center (SDSC) on a Sun Sparcstation 2. It can be run on any workstation that supports Unix and runs X Windows.

Several initial designs were considered for the Flagman. During experiments at SDSC, we considered the idea of having separate monitors run on each network machine. The monitors would handle the validation of problem solutions and the updating of shared global variables. No matter how the competition coordination was distributed, the results would have to be combined at some central point for consistency. We ultimately determined that the most flexible and reliable approach for the Supercomputing 93 competition was to have each computer send its results to one central monitor running on a workstation. The Flagman serves as this central monitor.

3.2 Network Interface

At Supercomputing 93, the Flagman used SCInet as the network interface. Since details of the network and the machines included on SCInet were not confirmed until very close to the start of the conference, we could not use specific information about the machines during most of development. For this reason, we chose to use generic network protocols. To obtain maximum flexibility, the network protocols allowing the nodes of the network to communicate with the Flagman were written using standard UNIX sockets. We established a minimum criteria which machines must satisfy to be able to participate in the FunRun. The necessary conditions were that the machine support the UNIX socket library and be "connectable" via the Internet. Our network library was ported to all of the machines of the FunRun with no modifications.

Figure 2: Flagman's workstation display. "Buttons" on the main window monitor which participants have logged in to which machines. Participants' windows display which puzzle pieces have been solved thus far. Timestamps in each show the validation time of the last puzzle piece solved for that participant.

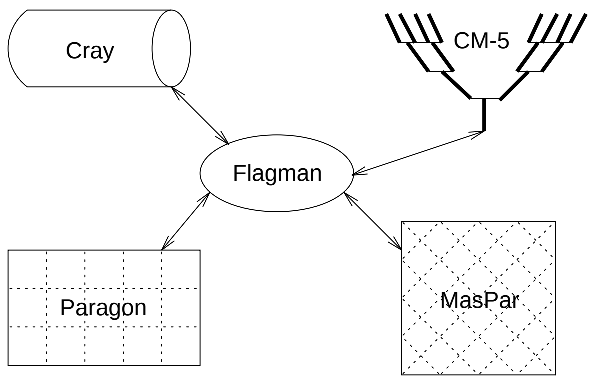

Figure 3: Star Topology of the FunRun Network.

Note that the Flagman and graphical user interface can survive the failure of any one of the node machines. Clearly however, the Flagman cannot survive the failure of the workstation on which it is executing. We used the star topology with the Flagman at the center for two reasons. At Supercomputing 93, the failure of a workstation executing the Flagman was relatively unlikely while temporary outages on SCInet would be more likely. The second reason is that each node machine need not know anything about other machines on the network. All of its correspondence takes place with the Flagman.

Our low level protocol combined with the star topology of the FunRun (Figure 3) provided the basic functionality required to validate and combine problem results. The implementation favored reliability and portability over other concerns.

3.3 Tasks

The machines chosen for solving the tasks in the competition were on the exhibition floor of Supercomputing 93, except for the MasPar[2]. The machines used were the Intel Paragon, Cray EL-98, Thinking Machine CM-5, and MasPar MP-2. These machines are different in architecture and support different computational paradigms. The Paragon is a space-shared MIMD massively parallel machine. The Cray EL-98 is a pipelined multiprocessor. The CM-5

[2]The MasPar was located in California and accessed remotely through SCInet.

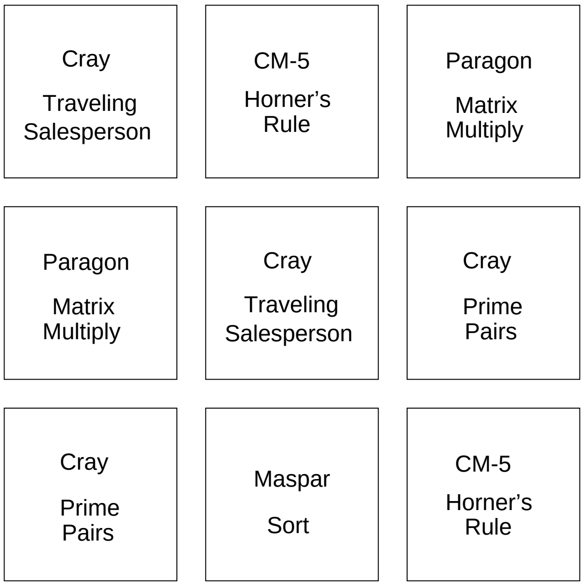

Figure 4: Distribution of problems to machines for the Supercomputing 93 competition.

supports both task-parallel and data-parallel programs in hardware, and the MasPar is a massively parallel SIMD machine.

For the competition, we enlisted the vendors' help in determining tasks that would execute well on each machine. For the Cray, a problem was chosen which could be highly vectorized. For the Paragon, a problem amenable to asynchronous parallelization was chosen, and for the CM-5 and MasPar, data-parallel programs were developed. Since there were four machines and 9 tasks, all machines except the MasPar were used to solve multiple tasks during the competition. Figure 4 shows the distribution.

Two problems were selected for the Cray: Traveling Salesperson and Prime Pairs. For Traveling Salesperson we did not ask for the shortest path, but for a path at least as short as could be found using a simulated annealing function which we provided. The simulated annealing approach uses an "uphill-climbing" strategy to find short paths. As the algorithm proceeds, it builds up arrays of cities and paths between them. The operations on these arrays are vectorizable, that is to say that operations can be done across the entire array rather than one element at a time. This fact made the problem an excellent candidate for execution on the Cray.

Prime Pairs is the problem of finding consecutive prime numbers which differ by two. We requested that the participant find the first prime pair above a certain number given as input. In order to present an unoptimized solution to the problem, the sample problem was given as a serial code for the Cray.

A SPMD matrix multiplication was performed on the Paragon. The values of the matrix were partitioned among the available processors, with each processor performing a matrix multiplication on the subset of data assigned to it. Intel supplied a kernel to perform matrix multiplication from their performance and test tools. Matrix sizes during the Supercomputing 93 competition were on the order of $5KX5K$.

Sorting of integers was performed on the MasPar. MasPar provided a SIMD sort which simultaneously performed the same operations on large data sets. Typical sorts performed during the competition were for 250000 to 500000 integers.

Horner's Rule was used to evaluate polynomials on the CM-5. Horner's Rule uses n multiplies and n adds in order to evaluate an nth degree polynomial. Due to the repetitive nature of the adds and multiplies, the CM-5 was used in SIMD mode in order to gain the best performance. Participants were asked to evaluate a 1000 degree polynomial.

3.4 Pacing the FunRun

A sample step of the competition – successfully solving a problem and obtaining a puzzle piece – proceeds as follows: The participant obtains input to a particular problem, either by solving an adjacent puzzle piece or by requesting it if it is the first piece. Using this input, the participant remotely logs in to the target machine and executes the program on the input. Compiled into the user's program is a call to *sendResult*, an object code utility supplied to the participants for the purpose of communicating with the Flagman.

SendResult is called when the execution on the target machine has completed and is ready submit a result. The result is sent to the Flagman with *sendResult*. The Flagman performs a validation step and sends the result of the validation (correct and validated, or incorrect and not validated) to the user's program on the target machine. The Flagman stores the results for each participant.

If the problem result is incorrect, the proposed solution is not validated. The participant is advised to try again and is given a different input data set. The new input penalizes the participant by forcing them to repeat the problem calculation. Rudimentary error messages are provided for some cases, such as if the puzzle piece earned was not adjacent to an existing accumulated puzzle piece. If the solution is correct and the earned puzzle piece is adjacent to an existing accumulated puzzle piece, the Flagman updates the graphical interface by adding the puzzle piece to the player's display, and provides the inputs to tasks whose puzzle pieces are adjacent to the piece just earned. The time the earned piece was validated is recorded and displayed as the most recent time stamp for that participant.

The FunRun terminates when time is called. The participant with the most puzzle pieces (or the most puzzle pieces whose last piece was awarded the earliest timestamp in case of a tie) "wins".

4 Beyond the FunRun

The format of the LeMans FunRun used at Supercomputing 93 allows a moderately sophisticated user to begin doing heterogeneous computing quickly. In our environment, many of the difficulties of real heterogeneous computing are hidden. In particular, the task partition and the allocation of tasks to machines is assumed to be fixed. In addition, standard communication protocols through *sendResult* are provided so that the programmer does not need to deal with data conversion between dissimilar machines. This was appropriate for the Supercomputing 93 setting since the requirement of more difficult partitioning, scheduling, or problem solutions would have reduced the number of participants who could commit to prepare for and enter the competition.

In a different setting, of course, it would be appropriate to provide a greater challenge for the participants and make the FunRun a more authentic representation of heterogeneous computing. In this section, we describe the degrees of freedom in the FunRun and discuss the expansion of the FunRun for use as an educational tool.

There are a number of ways that the FunRun could be expanded to be a more substantive competition. First, the problems solved by the participants could be considerably modified to provide more of a challenge. For Supercomputing 93, we put together a collection of unrelated tasks integrated by the puzzle format. For an expanded FunRun, a more authentic and unified heterogeneous problem could be used. In particular, a problem could be chosen where the participants are responsible for the partitioning, allocation of tasks to machines and scheduling during the competition. The problem would need to be structured so that milestones could be collected and displayed by the Flagman and so that for a fixed-time competition, partially solved problems could be compared and a "winner" could be determined. At Supercomputing 93, unoptimized sample code was provided for the tasks to make the participants' job more straightforward. A more extensive FunRun might offer sequential code or code developed for a single machine so that participants could verify their results during development.

The variables set in the Supercomputing 93 competition could be modified to produce an easier or more difficult FunRun. Time to complete the competition, size of the partitions which could be used on a space-shared machine, and the number of platforms which could and/or must be used are all variables which can be customized to vary the difficulty and length of the FunRun. If the network is known, the Flagman could use more sophisticated protocols and use the network and node topology more effectively.

When used in a classroom setting, the time restrictions, which affected the competition at Supercomputing 93, would not be present. Application programmers could interact with the students to provide experience with structuring a real heterogeneous problem, or the problem could be "synthetic" using the puzzle format to integrate pieces similar to the Supercomputing 93 competition. Student teams could take the time to decide which tasks would run on each machine and could have more time to develop optimized code to solve the problems. The teams could partition the tasks for different machines among themselves so that each member of the team becomes an "expert" on one machine when there are a moderate number of architecture types. Experiments to determine an optimal partition by generating different permutations of the possibilities could also be performed by the teams. The number of options available for students to explore can be restricted or expanded depending on how the competition is set up.

The FunRun aids in exposing students to difficulties in heterogeneous computing. Communication delays, data distribution, and task partitioning among the various machines all factor into the performance of a heterogeneous application and determine success in the FunRun.

Heterogeneous network computing is at once familiar and unfamiliar, involving the principles and practices from both parallel and distributed computing. The FunRun aims to provide a good introduction to heterogeneous network computing through an easy-to-grasp competition. Hands-on experience with software such as the LeMans FunRun can provide invaluable insight for programmers into the problems and potentials of this emerging and important field.

Acknowledgements

We would like to thank the San Diego Supercomputer Center for its assistance and support in developing the FunRun. We are grateful to Intel Corporation, MasPar, Thinking Machines Incorporated, Silicon Graphics and Cray Research Incorporated for their assistance in problem development and their generous assistance with the machines used in the competition. We would also like to give special thanks to Cherri Pancake, Ben Peek, Fairy Knappe, Dona Crawford, Len Wanger and the Dry Runners for their support and advice.