CCLRC / Rutherford Appleton Laboratory
Particle Physics & Astronomy Research Council
Starlink Project
Starlink System Note 23.3

SSN/23.3

T.M.Gledhill
C.A.Clayton
8 Jan 1997

Starlink Benchmarking Utility

Version v1.0

User's Manual

Abstract

The Starlink Benchmarking Utility provides a set of tools for investigating the performance of computer systems running astronomy data reduction software. This manual is intended for Starlink Site Managers and describes how to install and use the package.

Starlink Benchmarking Utility

Version v1.0

Contents

1 Introduction

System benchmarking is important not only in deciding what line to take in future hardware purchases, but also to check whether current systems are set up correctly, whether software is executing properly and as a tool to investigate configuration changes and tuning operations. Although there is no shortage of available benchmarks, many of them are unfortunately either very simplistic, perhaps running a piece of floating point Fortran code in a loop, or test a very specific area of system performance. In both cases the result is unlikely to give a true indication of how a real piece of software will perform.

The Starlink Benchmarking Utility is an easy–to–use set of tools for investigating the performance of computer systems when running astronomy data reduction software and currently includes benchmarks from the Starlink Software Collection and IRAF. The intention is to benchmark the sort of applications that are likely to be used by astronomers and hopefully to produce a performance estimate which is of more relevance to astronomers.

The three main aims of the Utility are:

- To provide an indication of how well a machine performs when carrying out tasks typical of astronomical data analysis.
- To check that all Starlink machines show consistent performance figures and that those figures are within expectations.
- To act as a performance diagnostic in system tuning operations, allowing the effects of hardware changes, such as adding extra memory or changing disk configurations, to be investigated.

The principal result produced by the Starlink Benchmarking Utility is the STARmark rating, which is described in Section 4. In addition, as of V1.0, an IRAFmark, based on the IRAF benchmarks, can also be produced. These ratings should provide convenient benchmarks for comparing the performance of different host systems. More detailed benchmarking results can also be produced and this is discussed in Section 5.

2 Installation

The package is currently supported on **Solaris** and **Digital Unix** platforms at version v1.0. Compressed tar files for each platform are available from the Starlink anonymous ftp account.

starbench_sun4_Solaris_v1.0.tar.Z[1]	(Solaris systems)
starbench_alpha_OSF1_v1.0.tar.Z[2]	(Digital Unix systems)

Installation of the package is straightforward and proceeds as follows:

1. Move to the empty directory in which you wish to install the package. A suitable place would be in the `/star/local` tree, *e.g.* `/star/local/starbench`. Space requirements are approximately 2MB for the compressed tar archive and 5.5MB for the installed package.

[1] ftp://starlink-ftp.rl.ac.uk/pub/tools/starbench_sun4_Solaris_v1.0.tar.Z
[2] ftp://starlink-ftp.rl.ac.uk/pub/tools/starbench_alpha_OSF1_v1.0.tar.Z

2. Obtain the compressed tar file, appropriate to your platform, from the Starlink ftp server either *via* the URLs given above or directly *via* FTP as shown below:

```
% ftp starlink-ftp.rl.ac.uk
(login as 'anonymous')
ftp> cd /pub/tools
ftp> bin
ftp> get starbench_***_v1.0.tar.Z
ftp> quit
```

3. Set the SYSTEM environment variable to identify your system.

   For Digital Unix systems:
   `% setenv SYSTEM alpha_OSF1`

   For Solaris systems:
   `% setenv SYSTEM sun4_Solaris`

4. Unpack the `tar` archive:

```
% zcat starbench_$SYSTEM_v1.0.tar.Z | tar xvf -
```

5. Execute the `mk` script. Note that the package will install into the current directory.

```
% ./mk install
```

3 Running the Benchmarks

The benchmarks are designed to be set up for automatic execution by the Unix `cron` daemon. This method of execution is preferred since, not only does it involve a lesser degree of user interaction, but it allows the benchmarks to be executed a number of times, at regular intervals, so as to build up a statistical base of results from which more reliable conclusions can be drawn.

3.1 Prior Requirements

The following setup is required in order to run the benchmarks.

- To run the benchmarks in the recommended fashion the user must have permission to run jobs using `cron` and `at`. This may necessitate placing the username in the files `/etc/cron.d/cron.allow` and `/etc/cron.d/at.allow`.
- The benchmarks themselves create large temporary files so that a minimum of 11MB of free space is recommended in the working directory (different from the installation directory).
- It is assumed that the **ussc** is installed under `/star` and that the user has access to it.

  If IRAF benchmarks are to be run, it is assumed that IRAF software is installed under `/iraf/iraf`. However, the environment variable `iraf` can be set to point to "non-standard" locations. For example, if IRAF is installed in `/star/iraf/iraf` then...

  `> setenv iraf /star/iraf/iraf`

  ...before starting up the benchmark package.

- The `tcsh` is used to time the benchmarks and is assumed to be in `/usr/local/bin`.
- The following packages are used by the benchmark suite and must be installed for the benchmarks to function properly:
  - **USSC:** KAPPA, PISA, SPECDRE, CCDPACK
  - **IRAF:** ccdred, daophot, images

3.2 Automatic Benchmarking – Recommended Procedure

The procedure for cron execution of the benchmarks will now be described, along with some guidelines intended to promote consistency in the results.

1. **Move to a suitable directory**

   During execution, the benchmarks create a number of temporary files, some of which can be quite large. These files are created in the current directory. The best policy is to move to a directory with a large amount of free space before issuing any benchmark commands; a minimum of 11MB is recommended. *In any case, the benchmarks cannot be executed whilst the current directory is the installation directory.*

   The location of the filesystems containing the current working directory, the benchmarking package, the Starlink software (ussc), and the iraf installation can all be expected to influence the benchmark results (this will be discussed later). It is suggested that, whenever possible, the current working directory should be on a filesystem which is **local to the machine being tested**. In addition, the filesystem should not be close to capacity, since full filesystems often experience reduced I/O efficiency. For more information on how disk I/O may affect the benchmark results, see Section 6.

2. **Source the startup script**

   To make the package available, source the startup script (`starbench.csh`) in the installation directory. For example, if the package was installed in `/star/local/starbench` then...

   ```
   host% source /star/local/starbench/starbench.csh
   ```

   An alias for this command could be defined in your `.tcshrc` script. This command sets up various environment variables, checks the software installation and also checks which packages (`ussc` and `iraf`) are installed on the system.

3. **Submitting the benchmarks for execution**

   The benchmarks are submitted for execution by the cron using a single interactive script, `submit`, which prompts for the necessary information. Here is a transcript of an example session:

   ```
   host% submit

    Submit
    ------
   ```

```
This is an  interactive script to submit benchmarks for execution
by the cron at regular intervals. In this way a benchmark profile
of the host machine can be built up.

The benchmarks will be run every hour. Give the number of minutes
past the hour to start each job. If you are running benchmarks on
several machines then it is a good idea to space them out.

Number of minutes past the hour [0] : 30

How many times should the benchmarks be run? The default is to run
them over a twenty-four hour period.

Number of times to run benchmarks [24] : 12

Give the name of a directory to hold the log files generated by
the benchmarks. The default is to  put them in the current work
directory.

Give a directory for the log files [.] :

The default is to just run the USSC benchmarks [ussc]. If you wish
you may run the IRAF  benchmarks instead  [iraf] or both  USSC and
IRAF benchmarks [both].

Specify which benchmarks to run [ussc] : both


...cron jobs started

warning: commands will be executed using /usr/local/bin/tcsh
job 793688940.a at Sat Feb 25 05:09:00 1995

...cron termination procedure queued

host%
```

The user is given the choice of when to execute the benchmarks (minutes past the hour) and how many times to execute them. When making this choice you should obviously avoid times when other heavy system processes, such as disk dumps or scratch disk purges, are executing. In addition, if benchmarks are being executed on more than one host, but are using the same disk space, then they should be spaced out in time so that they cannot interfere with one another. A reasonable option would be to benchmark 3 hosts at a time with executions separated by 20 minutes (*i.e.* 0, 20 and 40 minutes past the hour).

The user is given the choice of running the `ussc` benchmarks or the `iraf` benchmarks, or running both. *The default is to just run the ussc benchmarks.*

The `submit` script makes an entry in the user's crontab file causing the benchmarks to be executed every hour at the requested time. In addition, a script to terminate execution of

the benchmarks, after the required number of runs, and return the user's crontab file to its previous state is queued for future execution using the Unix `at` command.

Each time the benchmark suite is executed, the results are recorded in a log file (filetype `.log`) in the directory specified by the user. These log files contain the benchmark timings along with information on the host system characteristics. They are simple ASCII files and can be examined at any time.

3.3 Interactive Benchmarking

Although `submit` is the preferred command for running them, it is also possible to run the benchmarks on an interactive one-off basis using the command `bench`. Simply type `bench`, after sourcing the package startup script (previous section), and this will run the benchmarks and print the results out to the screen.

If the `bench` command line switch `-l` is used and the name of a log file given then the results will be sent to a log file rather than to the screen. If this option is used then the procedure could be backgrounded. This direct method of running the software will only produce a snapshot of the system and the results may not be as reliable as the multiple executions generated by `submit`. However, it is useful for getting quick results or when troubleshooting. For example:

```
host% bench -l quick-bench.log
```

Note that log files should have `.log` as their file type so that they can be recognised as such by the `scan` command.

The packages to benchmark can be selected using the `-p` command line switch, which takes the value `iraf`, `ussc` or `both`. For example, to only use the iraf packages:

```
host% bench -l quick-bench.log -p iraf
```

The default is to just use the `ussc` packages.

3.4 Processing the Log Files

The log files produced by each execution of the benchmarks can all be processed together using the `scan` command. This command accesses all of the log files in a given directory and sorts them according to host. Benchmarks executed on different hosts may log their results to a common directory and `scan` will sort them out, producing a results file for each host. For each host, average benchmark timings are computed and written to a file `<host>.bch`. These files are formatted lists of numbers and are largely for internal use within the package. If a `.bch` file already exists for a host, then `scan` will rename it and produce a new one.

The `scan` command then reads the `<host.bch>` files (one for each host) and calculates mean execution times and errors for each benchmark. These results are then compared with 'standard' measurements and a STARmark rating is calculated. If the IRAF benchmarks have been run, then an IRAFmark rating is produced. An example transcript follows:

```
host% cd /data/user/results
host% scan
```

```
 Scan
 ----

 This utility will scan through the benchmark logfiles created by
 the 'submit' or 'bench -l' commands to extract statistics and to
 calculate a  mean value and error for each benchmark.

 The performance is then compared with 'standard' results and the
 STARmark96  benchmark rating is calculated. If IRAF benchmarking
 was selected, then an IRAFmark96 rating will also be calculated.


 Give the directory path for the log files. The default assumes
 that they are in the current directory.

 Log files directory [.] : <RETURN>

 Benchmark Results
 =================

 Starlink Benchmark Utilities v1.0

 Hostname           : uhsul1
 User               : tmg
 Opsys              : SunOS 5.5
 Platform           : SUNW,Ultra-1
 Physical Memory    : 96.00Mb
 Online Processors  : 1
 Processor Speed    : 143MHz
 USSC version       : USSC176

 STARmark96 : 2.44 +/- 0.02  (based on 12 measurements)
 IRAFmark96 : 2.89 +/- 0.04  (based on 12 measurements)

host%
```

The results produced by the `scan` command can be sent to a log file *as well as* to the screen using the `-l` command line switch:

```
host% scan -l results.lis
```

The `scan` command can also produce extended results enabling a more detailed investigation of a system's performance. This facility is also very useful for checking that the benchmarks have executed correctly and that the results are not unduly influenced by any one application. Using these extended results is described in section 5.

4 The STARmark and IRAFmark Ratings

The STARmark and IRAFmark ratings are intended as estimates of how well a machine performs a sequence of astronomical data reduction tasks using Starlink and IRAF software. However, the applications that go to make up these two rating are totally different, including different mixes of CPU activity and disk I/O. It is therefore *not possible* to use STARmark and IRAFmark ratings to compare the performance of Starlink and IRAF software on a particular machine. They are, simply, measuring different things. The IRAF benchmarks have been included to provide another performance estimate for a particular hardware configuration, which will be more appropriate for sites running predominantly IRAF software.

Neither are the STARmark and IRAFmark ratings intended as definitive performance ratings for a machine. Indeed, they cannot be since they only test a limited subset of a machine's capability. Rather than trying to separate out and measure specific aspects of performance (such as CPU power, I/O efficiency, caching ability *etc.*) the current philosophy behind this package is to fold all of these factors into global performance estimates which, hopefully, will be of more relevance to the astronomer sitting in front of the screen. The STARmark and IRAFmark figures simply provide a convenient single-figure handle on these estimates.

Each benchmark is timed (using the shell `time` command) to give the "total CPU execution time" or **tcpu** for the benchmark. This is the sum of the user and kernel mode CPU times. In order to calculate the STARmark and IRAFmark ratings, the **tcpu** figures for each benchmark must be calibrated against a 'standard' result. In the case of the current package (v1.0) the standard is a 64Mb SPARC10 model 51 (RAL machine rlssp0). Comparison of the benchmark tcpu times on the machine being tested with those on the standard machine gives a **speed** rating for each benchmark (*i.e.* how much faster the benchmark runs on the machine being tested than on the standard machine), so that, for any particular benchmark:

$$\text{speed} = \frac{\mathsf{tcpu}_{\mathsf{rlssp0}}}{\mathsf{tcpu}_{\mathsf{machine}}}$$

Four benchmarks are currently incorporated in the STARmark rating, based on the KAPPA, PISA, SPECDRE and CCDPACK Starlink packages. The **speed** ratings for these benchmarks are averaged to form the STARmark rating.

$$\begin{aligned}\text{STARmark} = 0.25(&\text{speed(KAPPA)} + \text{speed(PISA)} + \text{speed(SPECDRE)} \\ &+ \text{speed(CCDPACK))}\end{aligned}$$

The error quoted on the STARmark figure is derived from the spread in the results from multiple executions of the benchmark package using the `submit` command. This spread in results is likely to reflect variations in load and activity on the system throughout the timespan of the benchmark executions. The quoted error therefore represents a lower limit on the real uncertainty which will include systematic effects (*e.g.* whether the working directory is on a local or remote filesystem) which may well dominate.

In a similar way, the IRAFmark rating is based on three IRAF packages: CCDRED, DAOPHOT and IMAGES.

$$\text{IRAFmark} = (\text{speed(CCDRED)} + \text{speed(DAOPHOT)} + \text{speed(IMAGES)})/3.0$$

The current list of expected STARmark and IRAFmark figures for various machines and configurations is available to Site Managers through the Starlink Forum System Managers' Conference, Note 4.0[3].

5 Extended Results

The `scan` command will print out more detailed results if given the `-f` command line switch. An example transcript is given below.

```
host% cd /data/user/results
host% scan -f

 Scan
 ----

 This utility will scan through the benchmark logfiles created by
 the 'submit' or 'bench -l' commands to extract statistics and to
 calculate a  mean value and error for each benchmark.

 The performance is then compared with 'standard' results and the
 STARmark96  benchmark rating is calculated. If IRAF benchmarking
 was selected, then an IRAFmark96 rating will also be calculated.


 Give the directory path for the log files. The default assumes
 that they are in the current directory.

 Log files directory [.] :


 Benchmark Results
 =================

 Starlink Benchmark Utilities v1.0

 Hostname           : uhsul1
 User               : tmg
 Opsys              : SunOS 5.5
 Platform           : SUNW,Ultra-1
 Physical Memory    : 96.00Mb
 Online Processors  : 1
 Processor Speed    : 143MHz
 USSC version       : USSC176

 STARmark96 : 2.44 +/- 0.02  (based on 12 measurements)
```

[3] http://rlsaxps.bnsc.rl.ac.uk/Forum/Systems-Management/4

```
IRAFmark96 : 2.89 +/- 0.04  (based on 12 measurements)

Average Load : 0.065 +/- 0.106

Benchmark  ucpu    kcpu       tpcu            Speed          Elapsed
---------  ----    ----       ----            -----          -------

      FFT  23.37   0.46  23.83 +/- 0.04    3.17 +/- 0.01   25.14 +/- 0.51

      SLA  12.06   0.02  12.08 +/- 0.10    1.47 +/- 0.01   12.42 +/- 0.18

    KAPPA   9.77   3.87  13.64 +/- 0.22    2.68 +/- 0.04   86.64 +/- 6.33

     PISA  10.34   2.40  12.75 +/- 0.23    2.43 +/- 0.04   48.47 +/- 1.12

  SPECDRE   7.87   3.81  11.68 +/- 0.18    2.47 +/- 0.04   35.01 +/- 0.54

  CCDPACK  13.32   5.33  18.66 +/- 0.18    2.17 +/- 0.02   67.20 +/- 1.94

   CCDRED  10.12   4.00  14.11 +/- 0.23    2.59 +/- 0.04  101.44 +/- 4.40

  DAOPHOT  16.05   3.91  19.96 +/- 0.17    1.55 +/- 0.01   37.18 +/- 1.49

   IMAGES  11.28   1.60  12.88 +/- 0.43    2.24 +/- 0.07   46.45 +/- 5.92
```

The full output shows the CPU execution times recorded for individual benchmarks, the **speed** ratings and the elapsed times. The nomenclature is as follows:

ucpu	:	Average CPU time (secs.) in user mode for each benchmark. This is the time the CPU spends executing the user's code.
kcpu	:	Average CPU time (secs.) in kernel mode for each benchmark. This is the time the CPU spends executing system instructions.
tcpu	:	Average Total CPU time (ucpu + kcpu) for each benchmark. The error on tcpu (secs.) is the standard deviation.
speed	:	The ratio of tcpu on the standard machine to tcpu on the machine being benchmarked. The errors on the 'speed' figures are simply the propagated tcpu errors.
elapsed	:	Average elapsed time (secs.) for each benchmark. The error is the standard deviation.

Since it is the four 'speed' figures for the KAPPA, PISA, SPECDRE and CCDPACK benchmarks that are averaged to give the STARmark rating, a full output can be used to check whether the computed STARmark is being biased by any particular benchmark. The same applies for the IRAFmark. The benchmark scripts do check for successful completion and flag log files with

problems so that `scan` will not use them. However, it is also a good idea to check the errors on the **tcpu** and **speed** figures to make sure that the results have not been influenced by a spurious result. If the errors appear high then the log files should be examined.

In addition to the 'astronomy' benchmarks, the full output also shows results for the FFT and SLA benchmarks. These benchmarks are purely CPU thrashers and do not do any disk I/O (although the FFT benchmark can page heavily on systems with less than 32MB physical memory). They can be useful in assessing to what degree the other benchmarks are likely to be influenced by disk I/O considerations but they are not incorporated in the STARmark or IRAFmark figures.

The individual benchmarks are described in Appendix A.

6 Disk I/O Effects

The package does not, at present, contain a disk I/O benchmark, although the inclusion of one has been considered at some length. The effects of disk I/O efficiency on overall performance are obviously important, especially in astronomy where large data files are routinely accessed. This section documents some of the problems inherent in producing a meaningful 'disk benchmark'.

First of all, one has to decide what to measure. The processes of disk reading and writing can be split into two parts. The first part, which we will call the 'internal' part, incorporates the calls to system I/O routines and the transfer of data to or from internal buffers and caches. The second part, which we will call the 'external' part, incorporates the actual transfer of information to or from a physical location on the disk. The efficiency of the internal part of disk I/O depends on the machine and operating system whereas the efficiency of the external part of disk I/O depends on the disk hardware and bus technology. These two aspects of disk I/O should really be benchmarked separately, however, there are too many possible hardware combinations here to produce a 'standard' benchmark result. Yet it is precisely the combined internal and external efficiencies which influence the overall disk I/O performance of the system.

The benchmarks in this package measure the **tcpu** (total CPU) times. In disk I/O processes it is only the internal (system) part of the I/O that is clocked by the CPU and hence appears in the **tcpu** figure. The external part of the I/O operation (physically reading and writing) only appears in elapsed time. The elapsed time that it takes a process to execute depends directly on what is going on on the rest of the system, in other words on the system load. Without knowing in advance how elapsed time on a particular machine scales with system load, the external portion of disk I/O cannot be benchmarked. One possible solution is to run a disk benchmark on a completely empty system where load is not a factor. However, even in this case the effects of system daemons and processes may need to be considered. Most background system daemon activity can legitimately be considered as part of the operating system, however sporadic activity caused, for example, by remote ftp connections or tftp requests from booting hardware, could influence the benchmarks.

In cases where a disk is not local to the system running the benchmarks then things are even more complicated, since the load on the machine serving the disk must be taken into consideration.

Whenever possible the benchmarks should be executed from a disk which is local to the machine being tested so as to minimise the effects of load on the file server on the benchmark results. In addition, filesystems which are close to capacity or which are heavily fragmented should be

avoided since they may suffer from reduced I/O performance. If it is suspected that the choice of disk is unduly influencing the benchmark results then the benchmarks should be run from several different disks and the results compared.

Finally, disk I/O benchmarks are heavily influenced by the effects of a disk cache. I/O to often-accessed small files can often be fulfilled from the cache and thus a benchmark involving such files would really be testing the speed at which your CPU can read blocks out of the cache and not the speed of your disk I/O subsystem. However, if your application actually does do a lot of reading and writing of small files, then the inclusion of the effects of the cache in your benchmark is valid. Also, the record length size plays a major role in performance measurements. Hence, the most reliable disk benchmark is your actual application. This is the approach taken in the present benchmarking suite. Real astronomical applications are used and thus they naturally include disk I/O elements of the relevant types and in the appropriate proportions.

7 Elapsed Time

The STARmark and IRAFmark ratings (Section 4) only consider the performance of a machine in terms of CPU time consumed and make no use of elapsed time. This is simply because elapsed time is highly dependant on machine load and it is not possible to produce reliable benchmarks even if one measures the machine load and attempts to account for it. Furthermore, it is usually not possible to measure elapsed time on an "unloaded" system since a networked Unix system is never idle.

Measurements of CPU usage, on the other hand, are highly repeatable and thus are suitable as a benchmark. They are also useful for identifying small changes in performance resulting from changes to system configuration.

Although elapsed time is difficult to interpret, it is of great interest to the interactive user who experiences elapsed rather than CPU time. The elapsed time is available in the individual log files produced by `submit` and `bench -l` and is output by the `scan -f` command. It can be particularly important when exploring the effects of disk location. For example, trials have shown that elapsed time for a benchmark run can increase by up to 500% when a remote disk is used rather than a local one whereas the increase in CPU time, although measurable, was small in comparison. Bearing in mind the difficulties inherent in benchmarking disk I/O, it is still often possible to extract useful information from the elapsed times and we suggest some guidelines for doing so:

- As described in Section 6 and above, elapsed time is highly dependent on system load in a way which is hard to quantify. Pick a period when the system should be relatively quiescent, *e.g.* overnight. Avoid other cron procedures such as disk dumps or scratch disk purges. Beware of http or ftp connections. When disk space (working directory, benchmark software or ussc) are on remote disks then activity on the disk server needs to be considered also.
- Run the benchmarks over an extended period of time using the `submit` command. For example, from midnight to 6am on several days of the week. Elapsed times are especially susceptible to sporadic system activity. Establishing a large base of measurements spread over time will smooth out these effects.

- Use the mean elapsed times produced by `scan -f` and check their standard deviations. If the standard deviation on a measurement appears large then check the benchmark log (`.log`) files for anomalous results. If you suspect that one particular measurement is unduly influencing the result then try removing the log file (*e.g.* by renaming the `.log` extension) and then run `scan -f` again.

8 Other Benchmarks

The benchmarks in this package are specifically intended to evaluate performance on systems principally running Starlink and IRAF software. On systems where other packages (*e.g.* AIPS) are heavily used then a benchmark specific to that package may be more appropriate. Such benchmarks may have been provided by the suppliers of the software.

A number of benchmarks based on IRAF tasks are included in this package at V1.0. However, a suite of more specialized IRAF benchmarks, which test the installation and performance of IRAF software, exists and was produced by NOAO. These benchmarks are described in the document *A set of Benchmarks for Measuring IRAF System Performance* written by Doug Tody which is available from the Starlink IRAF mirror as bench.ps.Z[4].

8.1 Systems running AIPS

A suite of benchmarks for systems running AIPS is available (the "Dirty Dozen Test" or DDT) which will produce an AIPSmark rating. The DDT suite is a reliable way of determining two things: firstly that a given AIPS installation gives accurate results, and secondly it is a measure of the performance of AIPS on the given system.

Full details of the AIPS DDT, including latest results, can be found at this URL[5]

9 Miscellaneous Issues

9.1 Changes to the ussc

The **ussc** is continually being updated and since the benchmarks use the installed copy of the collection, the tests being run will gradually change over time. Hence, it will be necessary to re-run the benchmarks every so often to measure the performance of machines with the current version of the **ussc**. The standard figures embedded in the benchmarking software will be updated once a year and the year identifier in the STARmark rating incremented to distinguish results from different incarnations of the software.

It might seem odd to use the everchanging installed copy of the software for the benchmarks rather than a self contained, static copy. However, the size of the benchmarking package would be enormous if it had to carry a private copy of the applications around with it. Moreover, we are interested in how fast new machines run our current release of the software, not a frozen copy, and hence despite the obvious drawback, we have elected to use the installed **ussc** for benchmarking.

[4] ftp://www.starlink.ac.uk/pub/iraf/iraf/docs/bench.ps.Z
[5] http://info.cv.nrao.edu/aips/ddt.html

9.2 New Versions of IRAF

Similar arguments to those in Section 9.1 apply with regard to IRAF software. The current V1.0 release of this package has been tested with IRAF V2.10.4. As new versions of IRAF appear, modifications to this package will no doubt be required.

9.3 Multiprocessor Machines

The benchmarks in this package do not test the benefit of multiprocessor machines. Since benchmark tasks all execute sequentially it is unlikely that they would take advantage of multiple processors.

9.4 Manufacturers' SPECmark Ratings

While the results from this package should be in broad agreement with manufacturers' SPECmark ratings, they will provide a more realistic performance estimate for Starlink machines. SPECmark ratings tend to indicate the *potential* that it is possible to realise with a machine rather than the performance that will actually be returned when running 'real' applications.

10 Acknowledgements

The production of this IRAF benchmarking software has been undertaken after discussions with Doug Tody, NOAO, the developer of IRAF. However, the benchmarking package has been developed completely independently of the IRAF team so that any errors in the package or spurious results that it might produce are wholly the responsibility of Starlink.

A Benchmark Descriptions

This appendix describes each benchmark and the operations it performs.

FFT:

A fortran program which runs a Fast Fourier Transform algorithm on a 1024x1024 element double precision array using KAPPA subroutines. This benchmark should test CPU floating point speed and the ability of the machine to handle large arrays. A considerable amount of virtual memory is allocated (approximately 32MB) which may result in substantial paging activity, depending on installed physical memory and other system processes. Fortran compiler optimisation flag -O is used.

SLA:

The SLA benchmark exercises each of the subroutines in the C implementation of the SLALIB Starlink subroutine package. This benchmark performs a substantial amount of floating point arithmetic with no disk I/O. C compiler flag -O is used.

KAPPA:

This benchmark runs a total of 22 KAPPA tasks sequentially (detail enhancement, image arithmetic, configuration change, compression and expansion, filtering and image statistics). Most of the image processing tasks use a 319x503 pixel real image. A substantial amount of disk I/O is done.

PISA:

This benchmark runs a modified version of the PISA demo script which locates and classifies objects on a test image and fits them with profiling functions. Again, a substantial amount of disk I/O is done.

SPECDRE:

This benchmark runs a modified version of the SPECDRE demo script which processes spectral data and performs several fits. Some disk I/O.

CCDPACK:

This benchmark runs a modified version of the CCDPACK demo which performs inter-image alignment, normalisation and mosaicing on test data. Disk I/O is quite intensive.

CCDRED:

This benchmark uses the CCDRED package in IRAF to create and process test CCD images. A substantial amount of disk I/O is done, and at least 10MB of free disk space is required.

DAOPHOT:

The IRAF DAOPHOT package is used to perform crowded field photometry of a simulated galaxy cluster image.

IMAGES:

The IRAF IMAGES package is used to perform image processing tasks on a test 512x512 CCD image.

B Changes

B.1 V0.8

The package was released at v0.8 in March 1995

B.2 V0.9

The following changes were made at v0.9:

1. All benchmark scripts now check for successful completion. Any anomalies are reported.
2. The output from the `scan` command has been improved. A new `-l` command line flag allows results to be logged to a file.
3. Benchmark scripts have been fixed to work with recent changes to the `ussc`.
4. The FFT benchmark has been re-written to use the PDA library.
5. The startup script now checks for the presence of benchmarked packages. Any absences are noted.
6. More system information produced as part of the benchmark headers.
7. Package startup is now achieved by sourcing a startup script `starbench.csh`.
8. Makefile updated to Starlink standard.

B.3 V1.0

The following changes were made at v1.0:

1. Three IRAF benchmarks based on the CCDRED, DAOPHOT and IMAGES package are included. These benchmarks use the locally installed version of IRAF except for modified login.cl and mkiraf scripts, which are included with this package.
2. The package startup script (starbench.csh) has been enhanced to check for the presence of `ussc` and `iraf` packages. It now prints out package availability.
3. A -p command line switch is added to the `bench` command, to facilitate running `ussc` and `iraf` packages. This is also used by the `submit` command.
4. The `scan` command now checks the benchmark version numbers in the log files before using them. It now calculates an IRAFmark where appropriate.
5. The SLALIB C benchmark uses customized comment-stripped code.