Hewlett-Packard Company
AlphaServer GS1280 7/1300

SPECfp_rate2000 = 78.0
SPECfp_rate_base2000 = 59.2

Hardware
CPU: Alpha 21364
CPU MHz: 1300
FPU: Integrated
CPU(s) enabled: 4 cores, 4 chips, 1 core/chip
CPU(s) orderable: 2 to 64
Parallel: No
Primary Cache: 64KB(I)+64KB(D) on chip
Secondary Cache: 1.75MB on chip per CPU
L3 Cache: None
Other Cache: None
Memory: 2GB per CPU; 256MB RIMMs
Disk Subsystem: AdvFS
Other Hardware: None

Software
Operating System: Tru64 UNIX V5.1B-1 + PK4
Compiler: Compaq C V6.5-011-48C5K
Program Analysis Tools V2.0
Spike V5.2 (510 USG)
HP Fortran V5.5A-3548-48D88
HP Fortran 77 V5.5A-3548-48D88
KAP Fortran V4.3 000607
KAP Fortran 77 V4.1 980926
KAP C V4.1 000607
File System: MFS, 8GB
System State: Multi-user

Notes/Tuning Information
Baseline C: cc -arch ev7 -fast -O4 ONESTEP
Fortran: f90 -arch ev7 -fast -O5 ONESTEP

Peak:
All use -g3 -arch ev7 -non_shared ONESTEP
except these (which use only the tunings shown below):
173.applu 188.ammp 191.fma3d
Individual benchmark tuning:
168.wupwise: kf77 -call_shared -inline all -tune ev67
   -unroll 12 -automatic -align commons -arch ev67
   -fkapargs=' -aggressive=c -fuse
   -fuselevel=1 -so=2 -r=1 -o=1 -interleave
   -ur=6 -ur2=060 ' +PFB
171.swim: same as base
172.mgrid: kf90 -call_shared -arch generic -O5 -inline
   manual -nopipeline -transform_loops -unroll 9 -automatic
Most benchmarks are built using one or more types of profile-driven feedback. The types used are designated by abbreviations in the notes:

+CFB: Code generation is optimized by the compiler, using feedback from a training run. These commands are done before the first compile (in phase "fdo_pre0"):

```
mkdir /tmp/pp
rm -f /tmp/pp/${baseexe}*
```

and these flags are added to the first and second compiles:

```
PASS1_CFLAGS = -prof_gen_noopt -prof_dir /tmp/pp
PASS2_CFLAGS = -prof_use_feedback -prof_dir /tmp/pp
```

(Peak builds use /tmp/pp above; base builds use /tmp/pb.)

+IFB: Icache usage is improved by the post-link-time optimizer Spike, using feedback from a training run. These commands are used (in phase "fdo_postN"):

```
mv ${baseexe} oldexe
spike oldexe -feedback oldexe -o ${baseexe}
```
Notes/Tuning Information (Continued)

+PFB: Prefetches are improved by the post-link-time optimizer Spike, using feedback from a training run. These commands are used (in phase "fdo_post_makeN"):

```bash
rm -f *Counts*
mv ${baseexe} oldexe
pixie -stats dstride oldexe 1>pixie.out 2>pixie.err
mv oldexe.pixie ${baseexe}
```

A training run is carried out (in phase "fdo_runN"), and then this command (in phase "fdo_postN"):

```bash
spike oldexe -fb oldexe -stride_prefetch -o ${baseexe}
```

When Spike is used for both Icache and Prefetch improvements, only one spike command is actually issued, with the Icache options followed by the Prefetch options.

**vm:**

- `vm_bigpg_enabled = 1`
- `vm_bigpg_thresh = 6`
- `vm_swap_eager = 0`
- `ubc_maxpercent = 50`

**proc:**

- `max_per_proc_address_space = 34359738368`
- `max_per_proc_data_size = 34359738368`
- `max_per_proc_stack_size = 34359738368`
- `max_per_proc_user = 2048`
- `max_threads_per_user = 4096`
- `maxusers = 2048`
- `per_proc_address_space = 34359738368`
- `per_proc_data_size = 34359738368`
- `per_proc_stack_size = 34359738368`

Portability: galgel: -fixed

Information on UNIX V5.1B Patches can be found at

Processes were bound to CPUs using "runon".