RC23276 (W0407-046) July 20, 2004
Computer Science

IBM Research Report

Power Measurement on the Apple Power Mac G5

Wes Felter, Tom Keller
IBM Research Division
Austin Research Laboratory
11501 Burnet Road
Austin, TX 78758

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Power Measurement on the Apple Power Mac G5

Wes Felter (wmf@us.ibm.com) , Tom Keller (tkeller@us.ibm.com)
IBM Austin Research Lab

1 Introduction

This report describes the instrumentation developed for the dynamic measurement of power and temperature for the Apple Power Mac G5. The G5 employs one or two 64-bit PowerPC 970 processors and runs Mac OS X, and 32-bit or 64-bit Linux. Apple has implemented a set of temperature, power and fan RPM sensors. OS X uses the sensors for fan speed control, in order to minimize the acoustic disturbance of a deskside system. The following metrics are obtained.

- Fans rpm
- Fans outlet temperatures
- Voltage to each 970 processor
- Current to each 970 processor
- Temperature of each 970 processor

Note: This report describes the first revision of the dual-processor 2.0 GHz Power Mac G5. By the time you read this, a newer liquid-cooled dual-processor 2.5 GHz model will be available.

The Austin Research Lab has a history of power instrumentation of systems [4 etc.] using external circuitry. For the first time we were given the opportunity to employ a desktop system already instrumented. While some Apple tools can display measurements from these sensors, there is no documented OS X API to access them. However, there is no such restriction under Linux. We used kernel modifications developed by IBM's Benjamin Herrenschmidt and made additional modifications to improve measurement fidelity. Since the only documentation we had was essentially marketing material and we had not developed the instrumentation circuitry ourselves, all the questions surrounding an unknown instrument set had to be resolved. For example,

- What is the accuracy and granularity of the measurements?
- How often are the measurements updated?
- How often could values be read with reasonable overhead?
- Are differences in readings due to regular error ranges in the instrumentation circuitry or differences in the components being measured?

The remainder of this report is organized as follows: a description of the hardware and software features relevant to power, and the measurement programs comprising the instrumentation. We next present benchmarks used for calibration/verification and their runs, followed by a discussion of our interpretation of the results. An IBM Confidential Appendix provides more detail about proprietary PowerPC 970 hardware features as well as source code for our benchmarks and tools.

We provide pointers to descriptions of hardware or software components that are publicly available elsewhere in an effort to reduce the length of this report.

[1] This work was partially supported by DARPA (Defense Advanced Research Projects Agency, U.S. Department of Defense) Power Aware Computing/Communication Contract F33615-00-C-1736 under Subcontract Agreement 400525-1.

2 Hardware Description

A 94-page description of the G5's hardware and software can be found at [1].

Apple's primary emphasis is on temperature: managing fans for minimal noise while providing dynamic cooling. Temperature and fans are not the emphasis of this report, however.

> "A common power management strategy is implemented across all Macintosh models. …To lower power consumption, heat generation, and fan noise, the Power Mac G5 computer incorporates an automatic power management technique called bus slewing. Bus slewing is designed to run at high processor and bus speeds and high voltage when the demand on the processor is high, and to run at low processor and bus speeds and low voltage when the demand on the processor is low. Switching between different processor/bus speeds and voltages is achieved by a gradual
>
> transition that does not impact system or application performance and operates seamlessly to the user. In slewing, the bus runs at half the speed of the processor.
>
> …
>
> In addition, the Power Mac G5 computer allows the user to control bus slewing mode. The options for specifying either high, reduced, or automatic processor and bus speeds are located at System Preferences>Energy Saver>Options; then select Automatic, Highest, or Reduced.
>
> If the Power Mac G5 computer detects a system temperature that is too high, due to high ambient temperatures or other factors, it will automatically enter bus slewing mode regardless of the selected setting." [1, p.17 of PDF version]
>
> "The Power Mac G5 system employs advanced thermal management to keep acoustic noise to a minimum…Temperature and power consumption are monitored by the operating system which communicates with the Fan Control Unit, which in turn controls and monitors fans. Each of the four thermal zones is equipped with its own dedicated, low-speed fans. Apple engineered seven of the nine fans to spin at very low speeds for minimum acoustic output. Using 21 different sensors, Mac OS X constantly monitors component temperatures in each zone, dynamically adjusting individual fan speeds to the appropriate levels for the quietest possible operation. As a result, the Power Mac G5 runs two times quieter than the previous Power Mac G4 enclosure." [1, p. 34]

We found no evidence of bus slewing during runs, implying that either Linux sets the energy saver options at "Highest" or that energy saver options are not implemented. While a number of idle modes are described in Apple literature, only a NAP mode was seen during our measurements. There is a significant difference in the power draw between IDLE-NAP and IDLE-NON NAP, on the order of 30 watts. There are three observed power states, the two mentioned, and RUN. Figure 1 shows examples of two processors in the three states.

2.1 Power/Thermal Measurement Hardware

Of the three types of measurements -- temperature, power and fan speed -- we will describe only the hardware necessary to determine the processor power: processor power supply voltage (V) and processor power supply current (I). The product of the two, V*I defines the instantaneous power, in watts W, of the processor. Figure 2 gives a cartoon view of how values for V and I are determined and read by the processor. A sense resistor is inserted in series into the supply line and the voltage drop across the resistor is used to obtain the current while the voltage of the regulated supply is used to obtain the voltage.

There is an Analog Devices AD7417 A/D converter [3] associated with each processor. The processor can read its 7417 through an on-board I^2C bus, via its Northbridge chip. The I^2C bus is capable of being read 20 times a second without perturbing performance (see driver description).

Calibration data for each processor is stored in a ROM. This is used to calibrate thermal values for the on-chip diode, min-max fan RPM, target temperatures, max power and other measures.

The setup drawn in Figure 2 is repeated for each G5 processor.

3 Linux Kernel Support Description

The Linux 2.6 kernel includes a driver called `therm_pm72.c` which controls the fans on the Power Mac G5 (internally, Apple calls this machine "PowerMac7,2"). This driver is based on open source code from Darwin and some guesswork. Although the purpose of the driver is to control fan speeds, it uses processor power consumption as input. The driver is structured as several control loops, one for each thermal zone; each loop reads the appropriate sensors, calculates new fan speeds, and sets the new fan speeds. In the stock version of this driver, these loops run once a second.

In the ARL-modified version of the driver the control loops still run once a second but the CPU voltage and current are read 20 times per second. We initially tried 50 times per second but the `kfand` kernel thread stopped responding, possibly due to livelock. This is not surprising, given that the I^2C bus is very slow and the G5 has many sensors.

As a side effect of controlling the fans, the therm_pm72 driver exports the following files in `/sys/devices/temperature`:

- `backside_fan_pwm`
- `backside_temperature`
- `cpu0_current`
- `cpu0_exhaust_fan_rpm`
- `cpu0_intake_fan_rpm`
- `cpu0_temperature`
- `cpu0_voltage`
- `cpu1_current`
- `cpu1_exhaust_fan_rpm`
- `cpu1_intake_fan_rpm`
- `cpu1_temperature`
- `cpu1_voltage`
- `detach_state`
- `devspec`
- `drives_fan_rpm`
- `drives_temperature`

Reading one of these files returns the most recent value calculated by the driver. Note: Reading these files does not cause the driver to read the sensors. The driver operates asynchronously from any access to these files. This means that a value read from these files may have been measured anywhere between 0 and 50 ms ago.

4 Tool Descriptions

We have written several power measurement tools for use on the G5. These tools are not publicly available, but we document them for reference.

4.1 g5_power

`g5_power` is a Python script that records voltage, current, computed power, and temperature for both processors over time and prints this data to standard output in self-documented tab-delimited format. The `interval` argument specifies the measurement interval in seconds; recommended values are between .1 and 2. The optional `iterations` argument specifies how many measurements to make; if omitted the tool will run until killed.

4.2 g5_power_cmd

`g5_power_cmd` is similar to `g5_power`, but it runs a specified command as a subprocess and takes measurements as long as the subprocess is running. It takes two arguments: `interval` and the command to execute.

Bug: The output of the command is intermixed with the output of g5_power_cmd, which can cause trouble. You may want to write a shell script wrapper which redirects the command's output to /dev/null.

Bug: You may see a traceback at the end of a run; this is harmless and can be ignored.

4.3 g5_power_mips_cmd

`g5_power_mips_cmd` is similar to `g5_power_cmd`, but it also outputs the MIPS of processor 0. MIPS is calculated using performance counters.

4.4 Graphing Power Data

Many graphing tools can be configured to ignore lines beginning with # so that you can directly graph the output of these tools.

Producing a power vs. time graph from this data using Ploticus[5] is as easy as

```
pl -prefab lines data=filename comment=# x=1 y=5 xlbl="Time (s)"
ylbl="Power (W)" title="title" pointsym=none
```

Likewise, this command produces a MIPS vs. time graph:

```
pl -prefab lines data=filename comment=# x=1 y=2 xlbl="Time (s)"
ylbl="MIPS" title="title" pointsym=none
```

4.5 Examples

Example invocations and output of the tools appear on the next page.

g5_power example:

```
$ ./g5_power .1 20
#arlx144        Fri May 28 21:56:37 CDT 2004    Linux arlx144 2.6.5-rc1-ames-G5-oprofile #9 SMP Tue Mar 30 17:13:28 CST 2004
ppc64 PPC970, altivec supported PowerMac7,2 GNU/Linux
#seconds cpu0_current cpu0_voltage cpu0_power cpu0_temperature cpu1_current cpu1_voltage cpu1_power cpu1_temperature
0.000052 22.949000    1.494000     34.285806  45.717000        16.357000    1.494000     24.437358  43.129000
0.133624 23.437000    1.494000     35.014878  45.835000        16.967000    1.496000     25.382632  43.129000
0.233627 23.559000    1.491000     35.126469  45.717000        17.089000    1.496000     25.565144  43.251000
0.333627 23.559000    1.491000     35.126469  45.835000        16.967000    1.494000     25.348698  43.251000
…
```

g5_power_cmd example:

```
$ ./g5_power_cmd .1 sleep 5
#arlx144        Fri May 28 22:02:06 CDT 2004    Linux arlx144 2.6.5-rc1-ames-G5-oprofile #9 SMP Tue Mar 30 17:13:28 CST 2004
ppc64 PPC970, altivec supported PowerMac7,2 GNU/Linux
#seconds cpu0_current cpu0_voltage cpu0_power cpu0_temperature cpu1_current cpu1_voltage cpu1_power cpu1_temperature
0.001641 23.071000    1.494000     34.468074  46.309000        16.357000    1.494000     24.437358  43.497000
0.119557 23.681000    1.491000     35.308371  46.427000        16.967000    1.494000     25.348698  43.619000
0.221541 23.803000    1.491000     35.490273  46.427000        17.211000    1.494000     25.713234  43.619000
0.328534 23.803000    1.491000     35.490273  46.427000        17.211000    1.494000     25.713234  43.619000
…
```

g5_power_mips_cmd Example:

```
$ ./g5_power_mips_cmd .1 sleep 5
#arlx144        Fri May 28 22:05:21 CDT 2004    Linux arlx144 2.6.5-rc1-ames-G5-oprofile #9 SMP Tue Mar 30 17:13:28 CST 2004
ppc64 PPC970, altivec supported PowerMac7,2 GNU/Linux
#['./g5_power_mips_cmd', '.1', 'sleep', '5']
#seconds MIPS     cpu0_current cpu0_voltage cpu0_power cpu0_temperature cpu1_current cpu1_voltage cpu1_power cpu1_temperature
0.004902 406.446933  22.949000 1.494000     34.285806  45.717000        16.479000    1.494000     24.619626  43.865000
0.120056 120.733193  23.681000 1.491000     35.308371  45.835000        17.211000    1.494000     25.713234  43.987000
0.220048 131.432219  23.925000 1.491000     35.672175  45.835000        17.333000    1.494000     25.895502  43.987000
0.320052 116.991124  23.925000 1.491000     35.672175  45.835000        17.333000    1.494000     25.895502  43.987000
…
```

5 Benchmarks

5.1 hot

Example:

```
$ ./hot 0 200000
```

`hot.c` executes the following loop using an 8KB array:

```
for(i=0; i<iter; i++) {
  for(j=0; j<SIZE; j++) {
          d += a[j]*i + a[j+1]*(i-1);
  }
}
```

It takes two arguments: `processor` specifies which processor to bind to, and `iterations` determines the number of iterations in the outer loop.

5.2 ops

Example:

```
$ ./ops 10
```

`ops` is a microbenchmark which runs 11 loops extracted from the LMBench [2] `lat_ops` benchmark, pausing between each one. These loops exercise different kinds of instructions and thus different functional units within the processor core.

5.3 lat_mem_rd

Example:

```
$ ./lat_mem_rd 1
"stride=128
0.00049 1.538
0.00098 1.521
0.00195 1.512
0.00293 1.510
…
```

The `lat_mem_rd` component of LMBench 3.0a3 creates a random-ordered linked list of cache lines and then traverses the list, causing random memory accesses. This is repeated for several working set sizes from 512 bytes up to a specified limit. If the working set size exceeds the cache size, `lat_mem_rd` will cause a cache miss on each access; this can be used to estimate the memory latency of the system. On most processors, `lat_mem_rd` will cause minimal power consumption since the processor is constantly stalled. The `len` argument is in MB.

5.4 MS_hot

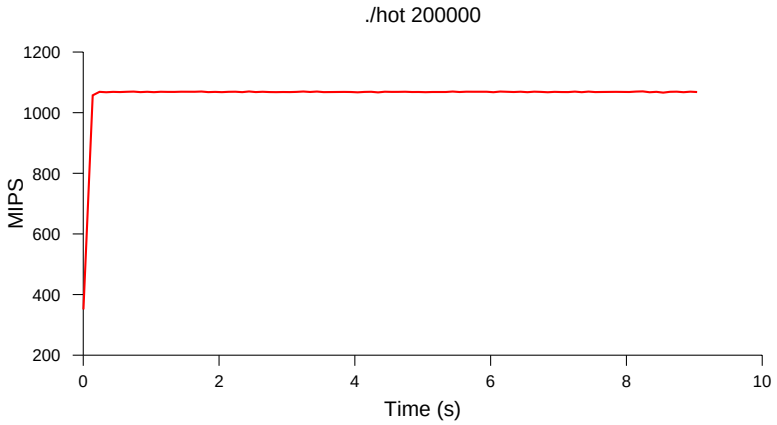
This benchmark executes exclusively out of the L1 cache, performing a tight loop of square root instructions. It was implemented by Karthick Rajamani and is available from the author. Source is not included in this document. The cases reported in this document were invoked by the command “ms T=4 b=16 e=16”.

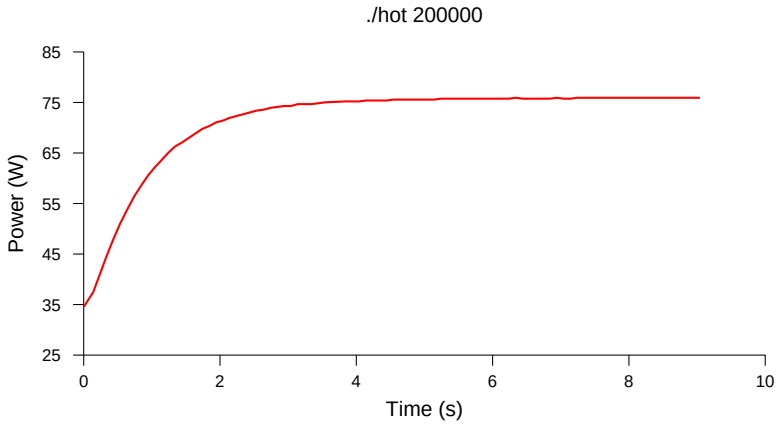
5.5 MS_cold

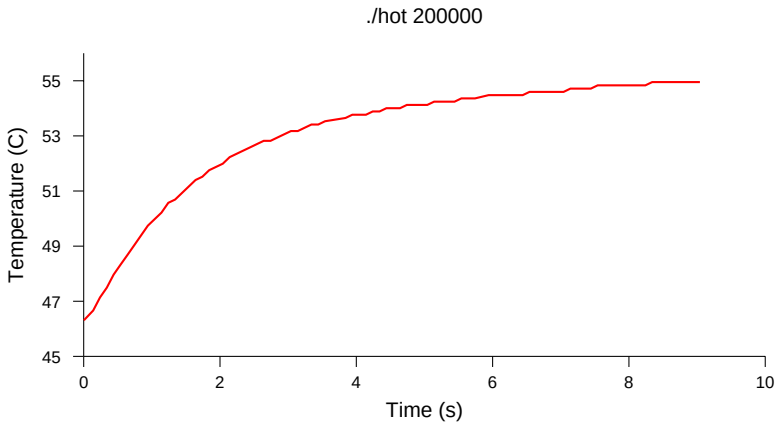
This benchmark populates an array of data greater than the size of the G5's L2 cache and then loads array values from a predefined and random ordering of addresses. Each operation is dependent upon the data of the previous operation, thus defeating pre-fetching. The intent of the benchmark is to stall the processor on RAM accesses to the greatest extent possible. It is invoked by the command "ms T=5 b=4096 e=4096". It was implemented by Karthick Rajamani and is available from the author. Source is not included in this document.

6 Benchmark Results

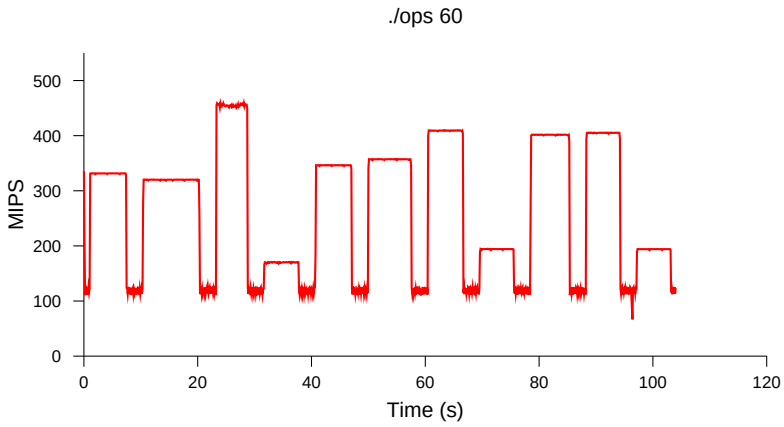
6.1 hot

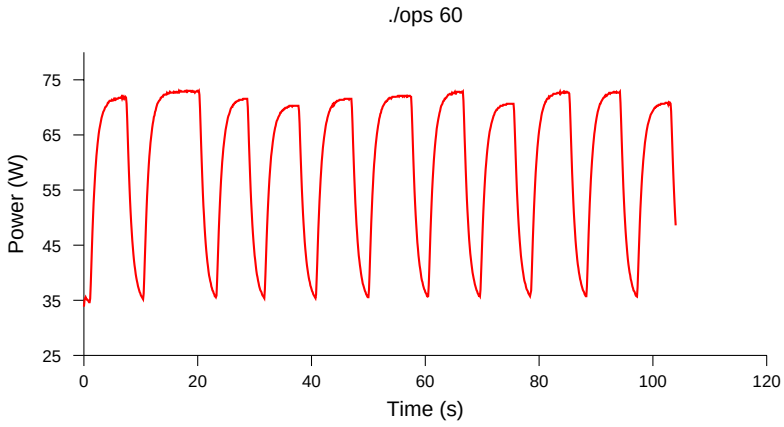


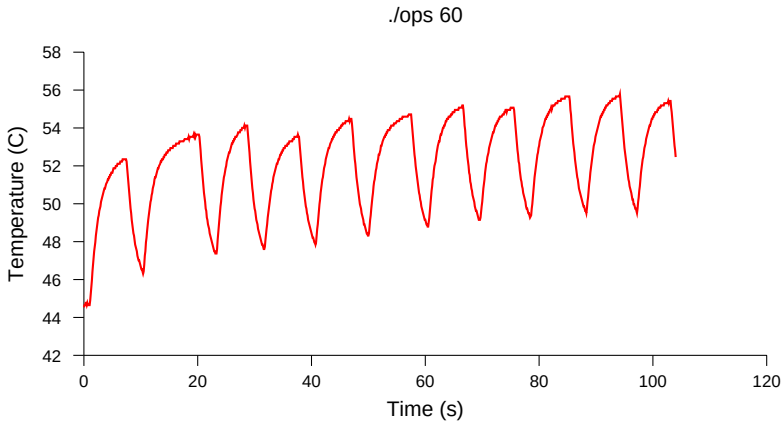




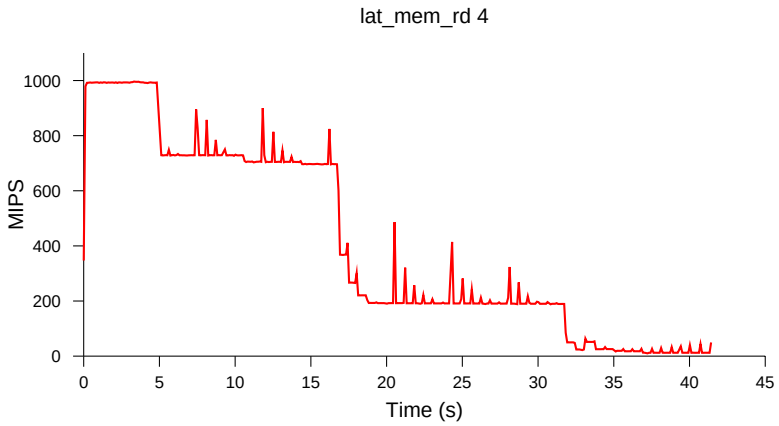
6.2 ops


./ops 60
MIPS
500
400
300
200
100
0
0
20
40
60
80
100
120
Time (s)

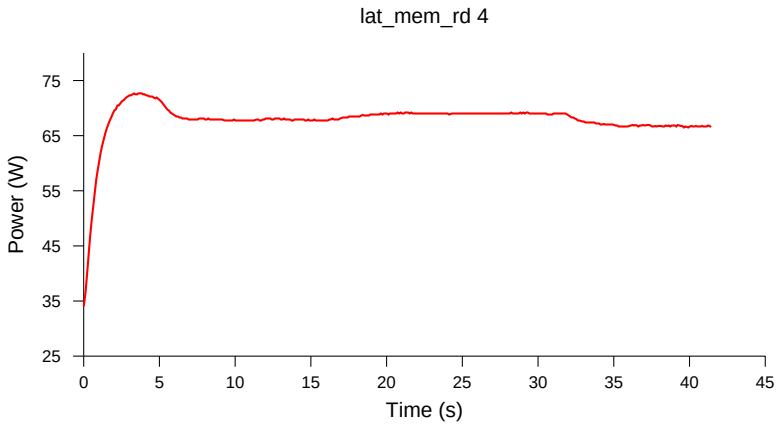

./ops 60
Power (W)
75
65
55
45
35
25
0
20
40
60
80
100
120
Time (s)

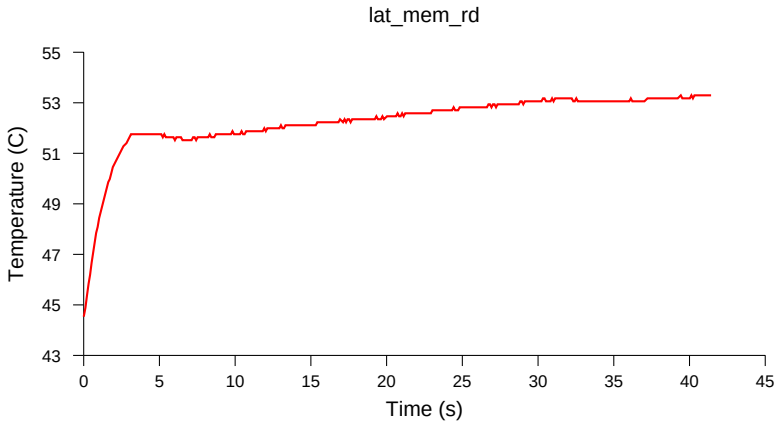

./ops 60
Temperature (C)
58
56
54
52
50
48
46
44
42
0
20
40
60
80
100
120
Time (s)

6.3 lat_mem_rd



Here four distinct phases are visible for working sets of 0-32KB, 32-256KB, 256-512KB, and 512-4000KB.





7 Discussion of Results

Figure 1 is used to illustrate and motivate much of the discussion in this section. Figure 1 displays measurements of power and temperature for two 970 processors in one of four states: NAP, IDLE, Cold Loop and Hot Loop. The loops are benchmarks described in a previous section as “MS_hot and MS_cold”. The script producing approximately 500 seconds of measures is

```
# Run each benchmark once in each of cpu 0 and cpu 1
# hot loop test does SQRT out of L1 cache
ms T=4 b=16 e=16 t=25 proc=0
sleep 5
ms T=4 b=16 e=16 t=25 proc=1
sleep 5
#
# cold loop test does random Loads out of RAM
ms T=5 b=4096 e=4096 t=25 proc=0
sleep 5
ms T=5 b=4096 e=4096 t=25 proc=1
echo "DONE"
```

The “proc=0” and “proc=1” arguments invoke a system call locking the process to the processor requested.

At time 0, both CPUs are in NAP mode. CPU 0 consumes 33 W and CPU 1 consumes 22 W of power. The temperature measurement for each chip is slightly higher than 40 degrees Celsius. When both processors have no work to do, both revert to NAP state. However, as soon as a single processor has work to do, its twin comes out of NAP state and into IDLE state. In the figure, CPU 0 is busy running a benchmark at 79 W while CPU 1 consumes 53 W at IDLE. Thus there are two “idle” states with significant differences in power consumption that we label NAP and IDLE.

The power consumption of a “busy” processor varies according to the activity of the processor. A “hot loop” benchmark and a “cold loop” benchmark were created to determine this difference in power consumption: 13 W for CPU 0 and 15 W for CPU 1. In both cases, Linux reports each processor’s utilization at 100%.

7.1 Processor-to-processor power variation

The data displayed in Figure 1 dramatically illustrates the difference in power consumption between twin processors in the same G5 system. In this case it is a near constant 11 watts across all states. The two simplest explanations for a consistent difference between processors are (1) different values in the sense resistors of the power measurement circuit or (2) manufacturing variations in the 9S2 technology, in particular leakage current variation. The authors favor the second explanation.

Power is calculated as the product of two voltage measurements: the voltage drop across a sense resistor and the supply voltage. In the case of the two CPUs the supply voltages are always nearly identical. The difference then rests in the measure of the voltage drop across the sense resistor. The two main contributors are the different resistances in each component, or the difference in the analog to digital conversion in the 7417 A/D converter. It seems less likely that a 7417 part would have such a large error in its A/D converters. Much more likely is different resistances. However, the difference is proportional to the 22 W and 33 W measures, meaning something like +/-5.5/27.5 tolerances, or +/- 20%, which seems excessive. The temperature measures also refute this theory. Note that the temperature of the napping 22W processor is significantly lower than the temperature of the napping 33W processor. If the power drawn by

each was identical, the temperatures should be identical. They are not. It's also improbable that the calibrated processor thermal diodes would coincidentally have similar errors in similar direction to the sense resistors.

In four pairs of processors sampled in G5's we found that (a) the processor wattages consistently maintained a constant difference (b) the reported cooler part was also the part with lower reported power and (c) the constant differences in power ranged from almost none to the 11 W covered by Figure 1.

Much more likely is the fact that the leakage current for each part is responsible for the difference. Intel parts and IBM parts in 130nm technology demonstrate considerable variations in leakage current, which is independent of operating frequency.

7.2 Increasing Temperatures Indicative of Leakage

Leakage current increases with the temperature of the part. This is evidenced in Figure 1 by the very slight positive slope in power during lengthy benchmark runs. The "hot loop" and "cold loop" benchmarks are performing constant work over many tens of seconds of time and therefore should be drawing constant power over time. They are not. The power consumption increases very slightly with time, as the temperature of the part increases.

7.3 Low-pass filter theory

On all other systems we have measured, a rapid transition in processor utilization (e.g. a square wave as produced by `ops`) produces a similarly rapid transition in power consumption. The Power Mac G5 displays different behavior: power rises and falls much more slowly than processor utilization. We believe this is due to an approximately 1/2 Hz analog low-pass filter in front of the current-measurement A/D converter. Such a filter would be useful when measuring power at the 1 Hz frequency that OS X uses, because it eliminates aliasing. Unfortunately, the filter skews results at our 20 Hz sampling rate. To accurately measure the power consumption of a given workload, it is necessary for the workload to be relatively stable and homogenous over a long time (~4 s) so that the filtered power converges with the instantaneous power.

When reading the power graphs in this paper or doing experiments, it is important to keep in mind that many of the samples are not accurate because of the low-pass filter. Effectively, it is impossible to accurately measure the power consumption of any highly variable workload.

7.4 Correspondence between power and temperature

Steady-state processor temperature is proportional to steady-state power consumption, but the large thermal inertia and long time constants of the heat sinks, temperature-dependent leakage power, and variable-speed fans make it difficult to predict temperature from power or vice versa.

An example of this can be seen in the `lat_mem_rd` benchmark; although the power is almost constant, there is an upward trend in temperature. Presumably, power is being dissipated into the heatsink faster than the fans are removing it, so the heatsink and processor gradually heat up. Presumably if this program ran long enough the system would increase the fan speed until the temperature reached equilibrium.

7.5 Correspondence between MIPS and power

Steady-state processor power consumption is proportional to processor MIPS for many kinds of workloads, although this is also complicated by temperature-dependent leakage.

The `ops` benchmark shows both where this relationship holds and where it does not. Most of the loops with higher MIPS also have higher power, except for the first two loops which have the highest power but lower MIPS than many of the other loops.

Measurements that include the kernel idle loop are a special case because power consumption is much lower when the processor enters nap mode.

8 Conclusion

Usage of the power and thermal measurement capability of the Apple G5 desktop system has been documented. The instrumentation is well suited for power measurements over intervals of many seconds, suitable for fan speed control. We have observed that there can be significant power consumption differences between 970 processors. We hope that interested readers of this report will extend this work to take further advantage of the first consumer desktop system which makes power and thermal data so readily available.

9 References

[1] Apple Power Mac G5 Developer Note, http://developer.apple.com/documentation/Hardware/Developer_Notes/Macintosh_CPUs-G5/PowerMacG5/index.html

[2] LMBench: http://www.bitmover.com/lmbench/

[3] Analog Devices AD7417 10-Bit Digital Temperature Sensor and Four Single-Channel ADCs data sheet, http://www.analog.com/UploadedFiles/Data_Sheets/48292559196992AD7416_7_8_f.pdf

[4] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, "The Case for Power Management in Web Servers". Appears in *Power Aware Computing*, Editors R. Graybill and R. Melhem, Klewer Academic Publishers, 2002.

[5] Ploticus: http://ploticus.sourceforge.net/

10 FIGURES

Figure 1. Example Measurements of PowerPC 970 Processor Swing

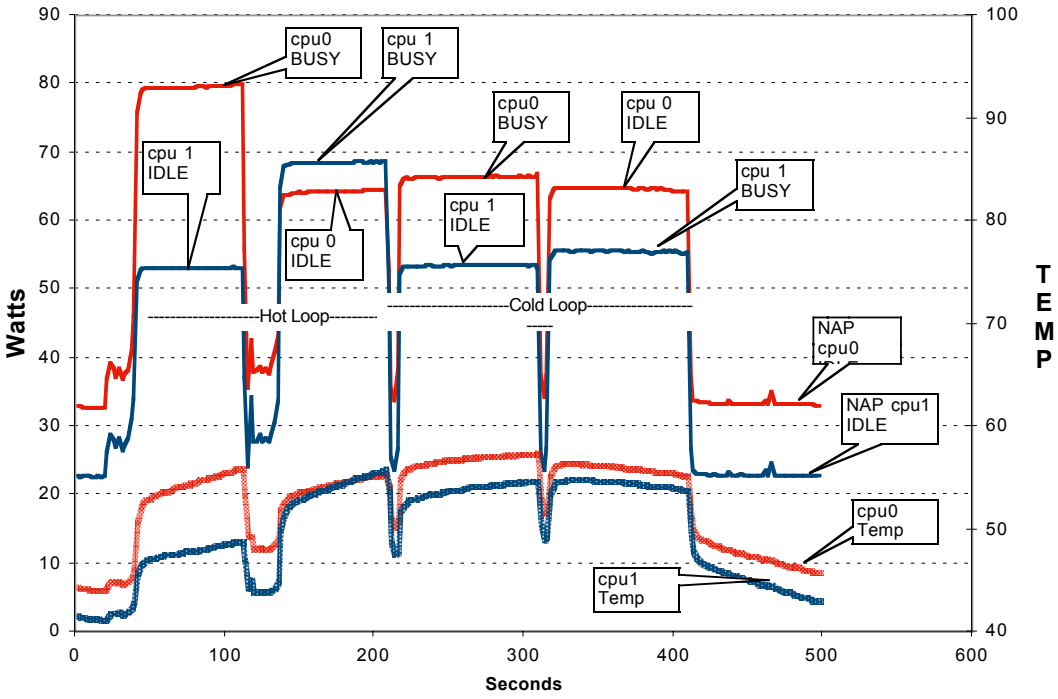


Figure 2: Hardware Instrumentation Cartoon