AMR5: Sorting 1TB with a mobile processor

Phillip Griffith

March 6, 2024

1 Introduction

This paper summarizes the configuration, benchmark steps, and results of a system that was built to improve upon the 1TB JouleSort benchmark (Indy Category). I use a desktop Mini PC equipped with an AMD Ryzen 7 5800U mobile CPU, and show an improvement over the current 2022 JouleSort world record [1].

My desktop Mini PC is a modified ACEMAGIC AMR5 Ryzen 7 5800U[2] equipped with two 2TB SSDs, 64GB of RAM, and a case fan. The system is powered by an external 65W power brick that comes standard with the AMR5. Using this hardware, I am able to sort a 1TB file, with 10^{10} records, in an average time of 1641 seconds, running at an average 32W (real power), consuming an average of 52,750 joules.

2 Hardware

The AMR5 Mini PC ran the JouleSort benchmark in a headless configuration, with no keyboard, mouse, or monitor attached. I used ssh to get a shell session over the network. The AMR5 is equipped with RGB LEDs, which I left connected.

2.1 Storage

I equipped the AMR5 with two 2TB Samsung 970 Evo Plus SSDs, for a total 4TB of sort storage. I used a single Samsung BAR Plus 64GB USB drive to hold the operating system and sort software.

2.2 CPU

My AMR5 Mini PC is equipped with a single AMD Ryzen 7 5800U CPU. AMD's data sheet for the 5800U rates it with a default TDP of 15W, and a configurable TDP of 10-25W, with 8 cores and 16 threads. The base clock is 1.9GHz, and the maximum boost clock is 4.4GHz. The 5800U is a Zen 3 chip, codenamed "Cezanne".

The AMD 5800U CPU was a very deliberate choice for the JouleSort benchmark. This CPU ranks fairly high on the Passmark Power Performance Ranking [3], and it's available at a reasonable price with the AMR5 Mini PC.

The AMR5 Mini PC has a performance knob that selects between silent, auto, and performance modes. Although technical documentation for the AMR5 is sparse, this knob seems to regulate the CPU's frequency, and hence the power draw, heat, and fan speed as well. I left this knob at its lowest setting of silent for the JouleSort benchmark.

The AMR5 Mini PC idles at about 9W.

2.3 Cooling

The AMR5 Mini PC ships with a CPU fan, but no case fan. To provide cooling for the SSDs, I enclosed them with copper heat sinks, and replaced one side of the case with a 140mm USB powered fan, rated at 1.5W. The power source for this fan is a USB port on the AMR5 Mini PC. Using this improvised case fan, the SSD temperatures stay about 9C below their rated maximum during the JouleSort benchmark.

2.4 Price List

All hardware components used in this system are commercially available at the time of this writing.

Part	#	Unit price	Total price
ACEMAGIC AMR5 AMD Ryzen 7 5800U Mini PC Barebone	1	$296.98	$296.98
Samsung BAR Plus 64GB - 300MB/s USB 3.1 Flash Drive	1	$11.99	$11.99
AC Infinity MULTIFAN S4, Quiet 140mm USB Fan	1	$15.99	$15.99
Crucial RAM 64GB Kit (2x32GB) DDR4 3200MHz CL22	1	$105.09	$105.09
icepc M.2 PCIE NVME 2280 SSD Copper Heatsink	2	$15.99	$31.98
Samsung 970 EVO Plus SSD 2TB NVMe M.2	2	$79.99	$159.98
watts up? PRO Power Analyzer (used)	1	$94.00	$94.00
System Total			$716.01

Table 1: Price List

3 Software

I installed Ubuntu Server 23.04 (GNU/Linux 6.2.0-31-generic x86_64) on the 64GB USB drive, and ran it from there. I used Nsort version 3.4.61 (Linux-X64) from Ordinal Technology to perform the sort.

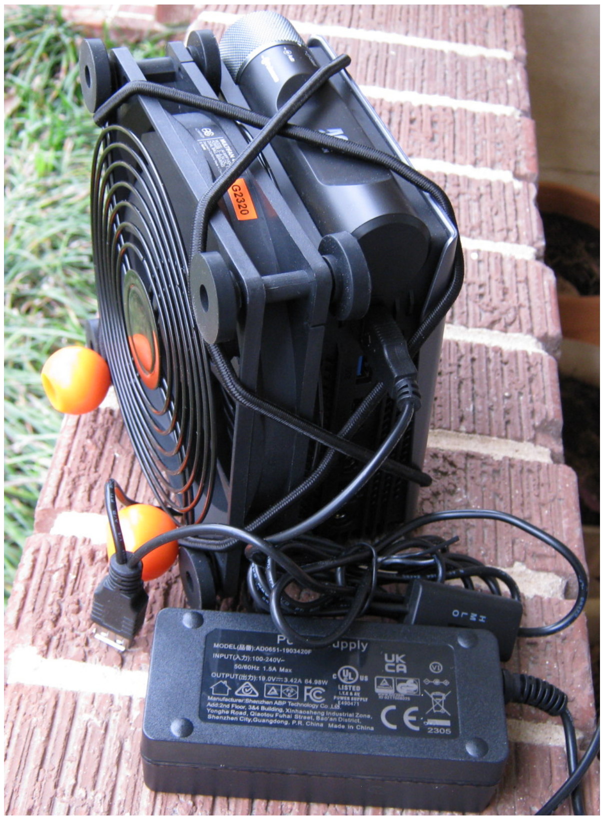

Figure 1: The AMR5 Mini PC with its power supply, and a case fan attached with bungee cords.

3.1 Storage Configuration

The two 2TB SSDs are joined together in a RAID 0 configuration to form a single 3.7TB volume, using mdadm(8), and mounted on */mnt/md0*. The RAID 0 volume is configured with the default 512KB stripe size.

This 3.7TB RAID 0 volume is used to hold the 1TB sort input, the 1TB sort output, and the sort work files.

3.2 Sort Filesystem

I chose the Flash Friendly File System (F2FS) to support the two SSDs in their RAID 0 configuration. I chose F2FS because it performs well in Phoronix benchmarks [4].

Sort benchmark rules do not allow data compression. Although F2FS supports compression, it's not enabled by default [5]. To comply with sort benchmark rules, I refrained from enabling compression when I created the F2FS filesystem.

```
phillip@amr5:~$ findmnt /mnt/md0 | cat
TARGET   SOURCE   FSTYPE OPTIONS
/mnt/md0 /dev/md0 f2fs
rw,relatime,lazytime,background_gc=on,⌋
↪   discard,no_heap,user_xattr,⌋
↪   inline_xattr,acl,inline_data,⌋
↪   inline_dentry,flush_merge,barrier,⌋
↪   extent_cache,mode=adaptive,⌋
↪   active_logs=6,alloc_mode=default,⌋
↪   checkpoint_merge,fsync_mode=posix,⌋
↪   discard_unit=block,memory=normal
```

Listing 1: F2FS mount options

3.3 CPU Frequency Scaling

CPU Frequency Scaling is a method of exerting software control over how fast the CPU runs. The Arch Linux wiki has a good overview at `https://wiki.archlinux.org/title/CPU_frequency_scaling`. CPU frequency scaling can influence performance per watt, and performance in the JouleSort benchmark.

For the AMR5's AMD CPU, I evaluated the amd_pstate and amd_pstate_epp drivers, before choosing the older acpi_cpufreq driver and the *powersave* governor to conserve energy, and to achieve the best results for the JouleSort benchmark.

4 Sorting

I used *gensort* to generate the 1TB ascii sort input file, and *valsort* to verify the sort output file. According to benchmark rules, I ran the sort five times against the 1TB sort input file, and recorded the results. Then I used gensort again to generate a second 1TB sort input file with skewed keys; then sorted the skewed sort input file, and reported those results.

```
phillip@amr5:~/64$ tb=`echo 10^10 | bc`
phillip@amr5:~/64$ echo $tb
10000000000
phillip@amr5:~/64$ ./gensort -a $tb
↪  /mnt/md0/sortin.txt
phillip@amr5:~/64$ ls -lh /mnt/md0
total 933G
-rwxrwxr-x 1 phillip phillip 932G Sep  9
↪  09:19 sortin.txt
```

Listing 2: Generating the 1TB input file with gensort

```
# clear pagecache, dentries, and inodes
sync; echo 3 | sudo tee
↪  /proc/sys/vm/drop_caches
date
start=$(date +%s)
echo begin sort
time nsort -processes=16 \
  -memory=60000M \
  -format=size:100 \
  -field=name:key,size:10,off:0,character
↪  \
  -key=key \
  -statistics \
  -in_file=/mnt/md0/skewed_sortin.txt \
  -out_file=/mnt/md0/sortout.txt \
  -temp=/mnt/md0
sync
date
echo end sort
end=$(date +%s)
echo "Elapsed Time: $(($end-$start))
↪  seconds"
echo "Begin 5 seconds cooling for the
↪  SSDs"
sleep 5
echo "Valsort beginning"
# validate output sort file
~/64/valsort /mnt/md0/sortout.txt
```

Listing 3: bash script used to run Nsort

5 Measurements

I measured the energy consumption of the AMR5 using a watts up? PRO power meter. The AMR5 Mini PC is plugged in to the power meter, which is, in turn, plugged in to the wall. The meter is connected to a separate monitoring computer via a USB cable. I followed the example of ELSAR [1] by using a publicly-available Python utility [6] that reads the meter's data from the /dev/ttyUSB0 port and saves them to a CSV file. Each reading is timestamped in the log file. Both computers use the systemd-timesyncd(8) service, synchronized to ntp.ubuntu.com.

Nsort statistics in Listing 4 show the CPU was 970% busy for the input phase, 608% busy for the output phase, and 764% busy overall. This means Nsort used an average 9.7, 6.08, and 7.64 CPU threads respectively.

5.1 Calculating energy consumption

For each run, the monitoring computer generates a separate watt log file, logged at one observation per second. I multiplied the average watts per second by the number of seconds between the starting and ending timestamps to arrive at the joules consumed during the sort. These calculations include the first and last seconds of the sort.

Run	Time(sec)	Power (W)	Energy (J)	Srec/J
1	1635	31.6	51,682	193,491
2	1582	32.1	50,849	196,661
3	1656	32.1	53,119	188,257
4	1695	32.2	54,649	182,986
5	1636	32.7	53,450	187,091
mean	**1641**	**32.1**	**52,750**	**189,697**
std dev	37	0.3	1,341	4,833
skewed	1677	30.7	51,561	193,945

Table 2: AMR5 Results

The mean power factor of the system was 60 during the sorts. The room temperature during the sorts was about 78F.

6 About the Author

I'm a retired IT worker with no computer science degree. My first exposure to sorting came during a computer science class in the late 1970s, when the instructor spoke approvingly of Syncsort on the IBM mainframe. My first job after college was in an IBM mainframe shop that ran Syncsort. I got to know Syncsort pretty well on that job.

During the mid 1990s, I moved from the mainframe to Solaris. I became a Linux hobbyist shortly afterward. Performance has been a longtime interest, but this project is my first attempt at publishing a performance benchmark.

```
3
Sat Sep  9 09:31:07 AM EDT 2023
begin sort
nsort -processes=16 -memory=60000M -format=size:100 -field=name:key,size:10,off:0,character
↪  -key=key -statistics -in_file=/mnt/md0/sortin.txt -out_file=/mnt/md0/sortout.txt
↪  -temp=/mnt/md0
Nsort version 3.4.61 (Linux-X64) using 2954M of memory out of 58G
Pointer sort performed Sat Sep  9 09:31:07 2023
          Input Phase       Output Phase         Overall
Elapsed   688.42             914.55               1602
I/O Busy  424.50    62%      913.74   100%        1338
Action  User   Sys Busy    User   Sys Busy    User   Sys Busy
sort    5950   729 970%    4626   938 608%   10576  1667 764%
  Rssmax     Majflt    Minflt  Sort Procs Aio Procs/QueueSize RegionKB
11830.50M      0/7      756755          16         0/8              512
File Name             ModeCntTran  Busy   Wait MB/sec  Xfers        Bytes     Records
Input Reads
  /mnt/md0/sortin.txt  dir 4x512k   62%  36.73   14541907349 1000000000000 10000000000
Temporary Writes
  /mnt/md0             dir 10x1m    74%    373   1455 954903 1000174428160
Temporary Reads
  /mnt/md0             dir 10x1m    72%    203   1095 954903 1000174428160
Output Writes
  /mnt/md0/sortout.txt dir 4x512k  100%    845   10941907349 1000000000000 10000000000

real        27m45.499s
user        176m16.671s
sys        28m50.270s
Sat Sep  9 09:58:55 AM EDT 2023
end sort
Elapsed Time: 1668 seconds
Begin 5 seconds cooling for the SSDs
Valsort beginning
Records: 10000000000
Checksum: 12a06cd06eeb64b16
Duplicate keys: 0
SUCCESS - all records are in order
```

Listing 4: Sample run of the sort script

References

[1] A. Kristo, P. Pillai, and T. Kraska. *Designing an energy-efficient, learning-enhanced algorithm to sort 1TB of ASCII data*. URL: http://sortbenchmark.org/ELSAR2022.pdf. (accessed: 09.09.2023).

[2] Acemagician. *Ace AMR5 AMD Ryzen Mini PC*. URL: https://www.acemagic.com/products/amr5. (accessed: 09.10.2023).

[3] PassMark Software. *PassMark CPU TDP Chart - Performance / Power of available CPUs*. URL: https://www.cpubenchmark.net/power_performance.html. (accessed: 09.09.2023).

[4] M. Larabel. *XFS / EXT4 / Btrfs / F2FS / NILFS2 Performance On Linux 5.8*. URL: https://www.phoronix.com/review/linux-58-filesystems. (accessed: 09.09.2023).

[5] Arch Linux Team. *F2FS - ArchWiki*. URL: https://wiki.archlinux.org/title/F2FS#Recommended_mount_options. (accessed: 09.09.2023).

[6] Y. Yoon and A. Kristo. *Watts Up? Pro/.Net meter logger*. URL: https://github.com/anikristo/wattsup. (accessed: 09.09.2023).