Performance evaluation for AAF Structured Storage files

Results collated by Phil Tudor, 5 April 2005, version 0.1

Introduction

This document presents the results of several pieces of work to evaluate the performance for reading & writing AAF Structured Storage files.

The results given here are:
Real-world AAF file writing speeds reported by end users
AAF SDK ScaleTest results
Evaluating use of OMCreateRawStorageCached

Real-world AAF file writing speeds reported by end users

Measurements by BBC, CNN, Fox

These results were obtained from operational systems and editing applications, with real-world projects.

Programme	Software	Hardware	Number of edits	Programme Duration	Number of tracks	Export settings	Export time
BBC							
Little Angels Prog 2 Behavioural Edit	Avid Adrenaline v1.3.5	HP W8000 crate 2x1.7Ghz Xeon	58	30’	1 Picture, 2 Sound	Include all tracks, Link to current media	“Almost instant”
Teen Angels Prog 1	as above	as above	805	56’ 30”	2 Picture, 8 Sound	as above	“30s at most”
BBC Sport							
Long Edit (with some Sapphire effects)	Avid Adrenaline v1.6. Windows XP SP1.	Xeon 3.06Ghz processor and 2GB of RAM		25’ 50”	1 Picture, 8 Sound		2s
Med Edit (multiple video effects)	as above	as above		5’ 34”	5 Picture, 8 Sound		4s
Short Edit (multiple video	as above	as above		2’ 55”	4 Picture, 4 Sound		3s

effects)							
BBC							
?	Quantel QeditPro						"take very little time to make"
CNN Post							
"So, after several attempts the CNN Post group could not make a Avid MediaComposer and FinalCutPro work together. The FCP kept crashing. They did get it to work with http://www.dharmafilm.com/sebskytools/ They said it was a reasonably complex file and there were no performance issues (that is when it worked)"							
Crawford Communications							
Input coming…							
Fox							
Trailer	Avid Adrenaline		128 (64 clips)	24' 30"	2 Picture, 8 Sound		2.75s
Hells Kitchen Presentation	as above		644 (199 clips)	1' 50"	2 Picture, 7 Sound		6.5s

AAF SDK ScaleTest results

Measurements by Stuart Cunningham, BBC

These results were obtained using the AAF SDK's ScaleTest, which creates AAF files of arbitrary size. Very large numbers of objects were tested, up to 40000 Mobs in a file.

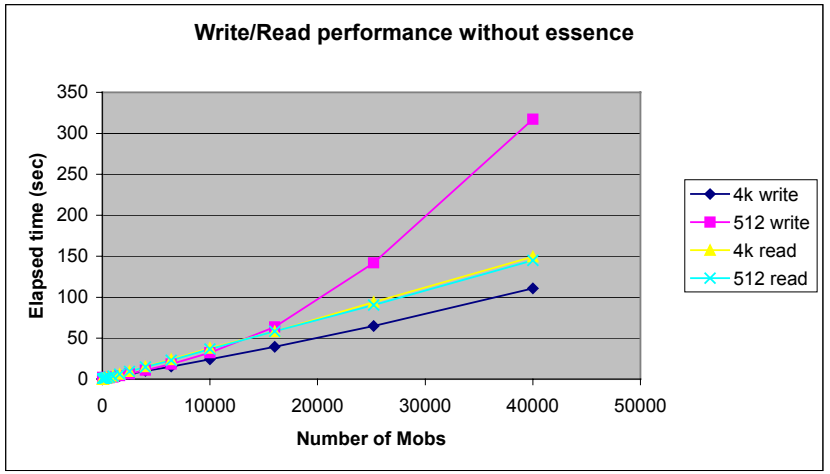
See next page for results. (Double-click to see whole worksheet)

Test platform was Linux using Schemasoft Structured Storage
Hardware was 2GHz Athlon with 1GB RAM

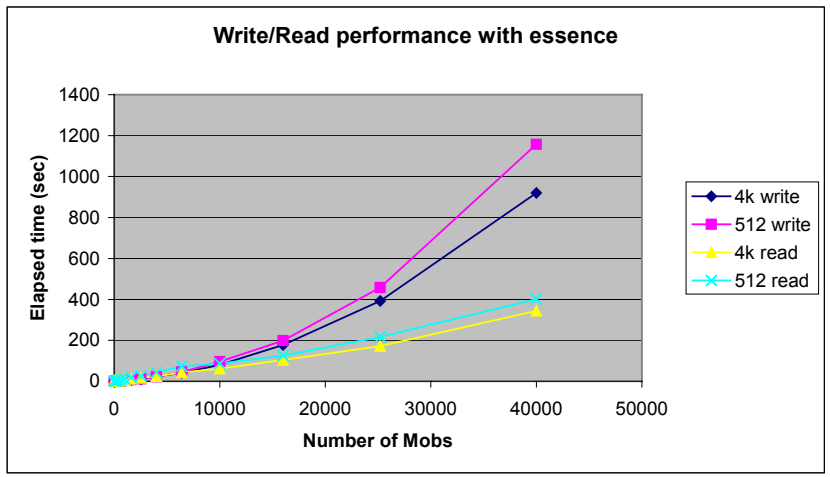
Observations:

- The performance of the AAF SDK with the SchemaSoft implementation has improved markedly since Dec 2003 [when BBC last performed these measurements]. A factor of 30 times speed improvement was observed for non-trivial files.
- In terms of scalability of the current SDK, a file containing 40,000 Mobs was created in 1min 50secs on a machine with a 2GHz Athlon CPU.
- A file containing 10,000 Mobs (the design goal for complexity for the AAF SDK) can be written in 24secs on a machine with a 2GHz Athlon CPU.
- Writing a 4k sector file is between 2 to 4 times faster than writing the equivalent 512 byte sector file. There was no tangible difference between 4k and 512 reading times.

Num sourc	Num Mobs	4k write	512 write	4k read	512 read
1	4	1.85	2.32	0.22	0.23
2	8	0.19	0.19	0.17	0.18
3	12	0.2	0.2	0.2	0.18
4	16	0.24	0.31	0.21	0.19
6	24	0.23	0.27	0.25	0.23
10	40	0.26	0.32	0.29	0.29
16	64	0.32	0.33	0.4	0.4
25	100	0.4	0.4	0.56	0.54
40	160	0.62	0.6	0.75	0.73
63	252	0.77	0.76	1.08	1.08
100	400	1.1	1.09	1.65	1.61
160	640	1.65	1.67	2.51	2.49
250	1000	2.58	2.64	3.88	3.73
400	1600	3.96	4.05	6.12	6
630	2520	6.14	6.52	9.45	9.36
1000	4000	10.05	10.89	15.22	14.78
1600	6400	15.24	18.3	23.79	23.15
2500	10000	24.15	32.41	37.96	36.6
4000	16000	39.52	63.72	58.1	58.49
6300	25200	64.83	141.78	94.08	90.42
10000	40000	110.69	317.25	149.27	145.05
16000	64000	207.34	752.78	4889.49	3325.7



Num sourc	Num Mobs	4k write es	512 write e	4k read es	512 read ess
1	4	0.18	0.96	0.81	0.95
2	8	0.19	0.2	0.26	0.34
3	12	0.2	0.21	0.25	0.53
4	16	0.22	0.24	0.28	0.7
6	24	0.24	0.25	0.33	0.48
10	40	0.29	0.31	0.42	1.95
16	64	0.36	0.39	0.62	1.48
25	100	0.47	0.53	0.82	1.55
40	160	0.66	0.83	1.19	2.62
63	252	0.96	1.38	1.78	3.71
100	400	1.44	1.69	2.99	4.69
160	640	2.44	2.56	4.44	7.47
250	1000	3.52	4.22	5.88	11.31
400	1600	5.95	7.44	11.96	20.08
630	2520	10.18	12.12	16.36	26.76
1000	4000	23.99	22.67	24.91	43.81
1600	6400	40.88	45.95	42.91	73.96
2500	10000	81.69	97.05	62.81	86.27
4000	16000	177.8	199.66	104.27	125.79
6300	25200	392.46	458.67	172.55	216.72
10000	40000	920.05	1157.65	343.94	401.54
16000	64000	FAILED	FAILED	N/A	N/A



Evaluating use of OMCreateRawStorageCached

Measurements by Jim Trainor, DiskStream

OMCreateRawStorageCached is an optional caching feature which is built into the AAF SDK. These tests evaluated how effective it was.

Results:
[from jim@diskstream.com] 7 March 2005
BTW recall that Tim warned that this does not show any improvement over Microsoft's native Windows SS implementation (I expect that it might even slow it down a bit).

[from jim@diskstream.com] 14 March 2005
In followup to the OM cache discussion at the engineering meeting:

I ran the ScaleTests, using 4000 frames, on Windows and Linux. In the windows case the test was run using both remote and locale storage.

In all cases the performance was lower when I activated the OM Cache. eli2aaf (the write test) ran vastly slower in some cases - up to two times slower. InfoDumper (the read test) was only a bit slower.

This tells me that the Schemasoft's structured storage implementation is already doing a good job caching IO requests. Either that, or I'm doing something wrong, but I don't think I am. The improvements Tim B. has reported must have been for truly horrible SS implementations (e.g. Microsoft's Mac impl).