Simple Access Interface

Motr Interfaces Workshop

5th Nov 2021

Andrew Lahiff (UK Atomic Energy Authority), SAI developed by Tyler Middleweek

What is SAI?

- ★ Simple Access Interface (SAI) provides a POSIX-like wrapper for Motr
- ★ Users can perform IO on objects via functions like:
  - ☆ open()
  - ☆ close()
  - ☆ read()
  - ☆ write()
- ★ No need to know or worry about the underlying Motr API
- ★ Makes porting of applications very simple

SAI highest level API

- Provides POSIX-like functions
  - Main difference: a sai_ObjectDescriptor is used instead of a file descriptor
  - Users don't need to supply or generate IDs

POSIX	SAI
int open(const char* *filename*, int *flags*)	sai_ObjectDescriptor *obj open(const char* *objname*, int *flags*)
read(int *fd*, void * *buf*, size_t *nbyte*)	read(sai_ObjectDescriptor *, void*, size_t)
write(int *fd*, void * *buf*, size_t *nbyte*)	write(sai_ObjectDescriptor *, void*, size_t)
close(int *fd*)	close(sai_ObjectDescriptor *)
remove(const char* *filename*)	delete(const char* *objname*)

Flags:

SAI_IO_READ -> O_RDONLY	Opens a file for writing.
SAI_IO_WRITE -> O_WRONLY	Opens a file for reading.
SAI_IO_READ_WRITE -> O_RDWR	Opens a file and allows it to be append.
SAI_IO_APPEND -> O_RDWR	Opens a file for appending.
SAI_IO_CREATE -> O_CREAT	Opens and creates the file if it does not exist.

Writes

Creating & writing a file

```c
#include <fcntl.h>
#include <unistd.h>

int main() {
    /* Open a file */
    int fd = open("myfile.txt", O_WRONLY | O_CREAT);

    /* Write some data */
    char stuff[] = "hello, world";
    write(fd, stuff, sizeof(stuff));

    /* Close the file */
    close(fd);
}
```

Creating & writing an object

```c
#include "sai_posix_wrapper.h"

int main() {
    /* Open an object */
    sai_ObjectDescriptor *obj = open("my_object.txt", O_WRONLY | O_CREAT);

    /* Write some data */
    char stuff[] = "hello, world";
    write(obj, stuff, sizeof(stuff));

    /* Close the object */
    close(obj);
}
```

- Writes limited to 32 blocks in one go
- Fix: put write inside a loop

Writing larger files

Creating & writing an object

```
#include <stdio.h>
#include "sai_posix_wrapper.h"

int main(int argc, char **argv) {

        const char *file_to_copy   = argv[1];
        const char *obj_to_copy_to = argv[2];

        int block_size = 4096;

        FILE* file = fopen(file_to_copy, "rb");
        sai_ObjectDescriptor *obj = open(obj_to_copy_to, O_WRONLY | O_CREAT);

        size_t size;
        char *buffer = malloc(block_size);
        while ((size = fread(buffer, 1, block_size, file)) > 0) {
                if (sai_write(obj, buffer, size) <= 0) {
                        free(buffer);
                        close(obj);
                        return 6;
                }
        }

        free(buffer);
        fclose(file);
        close(obj);

        return 0;
}
```


Reads

Reading a file

```c
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

int main() {
    /* Open a file */
    int fd = open("myfile.txt", O_RDONLY);

    /* Read some data */
    char buffer[13];
    read(fd, buffer, sizeof(buffer));

    /* Close the file */
    close(fd);

    /* Print the data */
    printf("%s\n", buffer);
}
```

Reading an object

```c
#include "sai_posix_wrapper.h"

int main() {
    /* Open an object */
    sai_ObjectDescriptor *obj = open("my_object.txt", O_RDONLY);

    /* Read some data */
    char buffer[13];
    read(obj, buffer, sizeof(buffer));

    /* Close the object */
    close(obj);

    /* Print the buffer */
    printf("%s\n", buffer);
}
```


Deletes

Deleting a file

```c
#include <stdio.h>

int main() {
    remove("myfile.txt");
}
```

Deleting an object

```c
#include "sai_posix_wrapper.h"

int main() {
    delete("my_object.txt");
}
```


Not just strings

Write model

```
bool model_write_to_obj(const Model *model, const char *filename) {

  sai_ObjectDescriptor *obj = open(filename, O_WRONLY | O_CREAT);
  if (obj == NULL) {
    printf("failed to write to model file\n");
    return false;
  }

  write(obj, &model->ball_start_mass, sizeof(model->ball_start_mass));
  write(obj, &model->grav_acc, sizeof(model->grav_acc));
  write(obj, &model->ball_start_pos, sizeof(model->ball_start_pos));
  write(obj, &model->ball_start_vel, sizeof(model->ball_start_vel));

  close(obj);

  return true;
}
```

Read model

```
bool model_read_from_obj(Model *model,  const char *filename) {

  sai_ObjectDescriptor *obj = open(filename, O_RDONLY);
  if (obj == NULL) {
    printf("failed to read model file\n");
    return false;
  }

  read(obj, &model->ball_start_mass, sizeof(model->ball_start_mass));
  read(obj, &model->grav_acc, sizeof(model->grav_acc));
  read(obj, &model->ball_start_pos, sizeof(model->ball_start_pos));
  read(obj, &model->ball_start_vel, sizeof(model->ball_start_vel));

  close(obj);

  return true;
}
```


Tracking the state of files

- ★ SAI needs to store IDs & file size for each object somewhere
- ★ Option 1 – reference files (default)
  - ☆ A `<object name>.sai` file is created for each object written by SAI
  - ☆ Store object IDs & size

```
$ cat model_example_data.txt.sai
5644123107195889027 0 223890
```

  - ☆ These files are needed to read objects
- ★ Option 2 – use the Motr key-value store (in progress)
  - ☆ Object ID & size stored in an index

Configuration

- Resource files
  - Looks for resource file of the form
    `/home/users/jusers/<username>/sage/.c0appz/c0cprc/<hostname>`
  - Created by “make sagercf” from Clovis sample apps
- Block sizes
  - Default hardwired 4 kB block size
  - Users can override this by defining the environment variable **SAI_BLOCK_SIZE**
  - May add configuration file option as well
- When key-value store is used instead of ancillary reference files the following environment variables need to be defined:
  - **SAI_INDEX_ID_HI, SAI_INDEX_ID_LO**

SAI lower level API

Writing an object

```c
#include "sai_object.h"

int main() {
    /* Open an object */
    m0_uint128 id = sai_generate_id("my_object.txt");
    sai_ContextMotr *motr = sai_ctx_motr_get();
    sai_object_create_with_id(motr, id);
    sai_Object obj;
    sai_object_init(&obj, motr);
    sai_object_open_with_id(&obj, id);

    /* Write some data */
    char stuff[] = "hello, world";
    sai_object_write(&obj, stuff, sizeof(stuff), 0);

    /* Close the object */
    sai_object_close(&obj);
    sai_end(motr);
}
```

Create unique object ID from object name

Get a Motr context

Create object

Initialises an allocated sai_Object structure

Opens a sai_Object

Closes the sai_Object

Deallocates the Motr context

sai_ContextMotr
- Motr client, Motr container, …

sai_Object
- Associates a Motr object with a unique ID

sai_ObjectDescriptor
- sai_Object
- Object size
- Current offset
- File handle for the ancillary reference file

Reading an object

```c
#include "sai_object.h"

int main() {
    /* Open an object */
    m0_uint128 id = sai_generate_id("my_object.txt");
    sai_ContextMotr *motr = sai_ctx_motr_get();
    sai_object_create_with_id(motr, id);
    sai_Object obj;
    sai_object_init(&obj, motr);
    sai_object_open_with_id(&obj, id);

    /* Read some data */
    char stuff[13];
    sai_object_read(&obj, stuff, sizeof(stuff), 0);

    /* Close the object */
    sai_object_close(&obj);
    sai_end(motr);

    /* Print the buffer to the screen */
    printf("%s\n", stuff);
}
```

Performance: comparison to Clovis sample apps

2 MB file, 4 KB block size

		Clovis (s)	SAI (s)
Copy to	Open	3.6	4.7
	Copy	7.4	7.2
Copy from	Open	3.5	4.8
	Copy	1.6	1.8

65 MB file, 64 KB block size

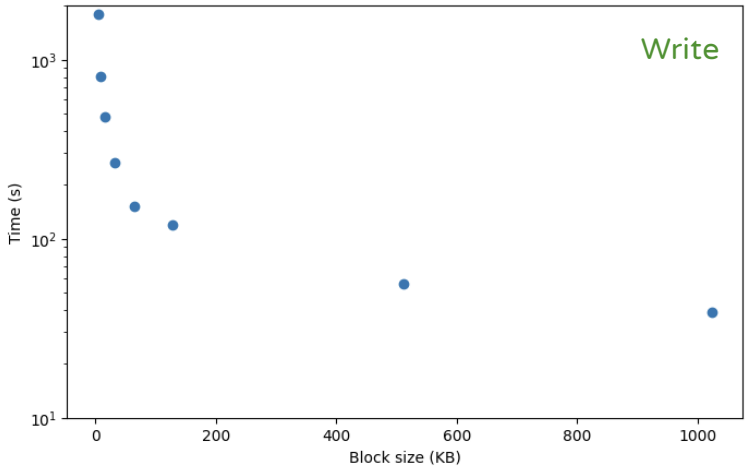
		Clovis (s)	SAI (s)
Copy to	Open	3.6	4.7
	Copy	11.8	11.7
Copy from	Open	3.6	4.8
	Copy	5.6	5.4

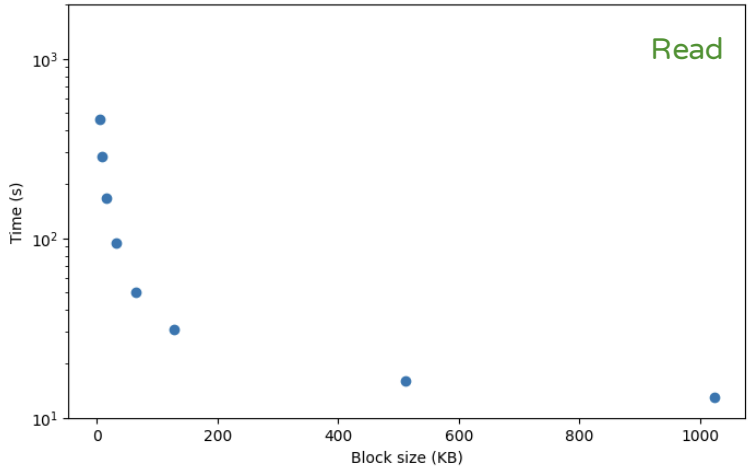
512 MB file, 1024 KB block size

		Clovis (s)	SAI (s)
Copy to	Open	3.6	4.8
	Copy	47.7	42.8
Copy from	Open	3.7	4.6
	Copy	14.9	13.5

Performance: block size

Time to write & read 512 MB file as function of block size





Thank You

andrew.lahiff@ukaea.uk