General LIBMSR Use

Libmsr Version 0.2.1

Before you can use any of the LIBMSR functions, you must call the init functions.

Initialize LIBMSR

1. Call init_msr, if it returns -1 there was an error opening the msr_safe or msr kernel modules.
2. If you are using RAPL, you also need to call the rapl_init function. If this returns -1 then RAPL is probably not supported on your architecture.

Initializing

```
// Passing this to rapl_init by reference gives you one way to access the rapl data

struct rapl_data * rd = NULL;

// You can pass this to rapl_init to enable/disable MSRs in case the

// auto-detect missed anything

    uint64_t * rapl_flags = NULL;
// These are both optional. See the RAPL section for more details.

    if(init_msr())

    {

        return -1;

    }

// If you don't need rapl data or custom flags settings, these can be NULL.

    if (rapl_init(&rd, &rapl_flags))

    {

        return -1;

    }
```

Finalize LIBMSR

Before you return from main, you will want to call finalize_msr. This will close file descriptors, free memory, and do other various cleanup tasks. Often times, after you are done doing any testing you will want to change the RAPL limits back to their default values. This can be done by calling the set_to_defaults function. Be sure to call this before any finalize functions.

Finalizing

```
    set_to_defaults(); // This will set all RAPL limits to their default values

finalize_msr(); // This will restore MSRs to their prior values
```

Viewing Available MSRs

There are functions that query available RAPL MSRs and performance counters: print_available_counters() and print_available_rapl(). These are a work in progress so they may miss registers that exist on your system.

Accessing Raw MSR Data

If you need the raw data from MSR's there are functions in each domain that you can use. These functions are generally use the format of "thing_that_i_want_storage". For example, if I want the fixed counters MSR data I would use "fixed_ctr_storage". These functions differ between domains so be sure to check the header files, there are plans to make these more uniform in the future. Also note that you have to take care of any initialization before this will work.

Using Storage Functions

```
// Get fixed counter MSR data
struct ctr_data *c0, *c1, *c2;
fixed_ctr_storage(&c0, &c1, &c2);
int i;
for (i = 0; i < totalThreads; i++)
{
// display the raw data
fprintf(stdout, "%lx", ctr->value);
}
```

Related articles

- PCI Configuration Registers (CSRs)
- The Batch Interface
- RAPL
- Performance Counters
- General LIBMSR Use