Tools and GPU Runtimes Working Group Outbrief

EXASCALE COMPUTING PROJECT

John Mellor-Crummey (Lead and Scribe)

23 June 2022

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

Tool Concerns

- **Initialization**
  - when should a tool be initialized?
    - before main
    - before most constructors
    - before offloading to GPU
    - before creating threads that need to be monitored
- **Threads**
  - programming models and runtimes have threads for many purposes
    - MPI progress thread, OpenMP workers, runtime support for managing GPU offloading
  - some threads exist only to support tools
    - monitoring kernel launches, reporting asynchronous events, recording activities
  - not all of these threads should be monitored

Potential Approaches

- **Have each library maintain state about each thread it is trying to create + an inquiry API**
  - where is the state maintained?
  - how does a tool use the API to access that state?
- **Pass an attribute to pthread create that indicates the role of the thread**
  - information at the right time. arguments with a standards committee would be endless
- **Assign each thread a name based on its role: pthread_setname_np, pthread_getname_np**
  - https://man7.org/linux/man-pages/man3/pthread_setname_np.3.html
  - name may only be assigned after a thread is created
- **Metadata in compiled code**

The Most Promising Approach: Metadata in Compiled Code

- **ELF Notes**
  - See : https://man7.org/linux/man-pages/man5/elf.5.html
  - ELF notes allow for appending arbitrary information for the system to use.

```
typedef struct {
    Elf64_Word n_namesz;
    Elf64_Word n_descsz;
    Elf64_Word n_type;
} Elf64_Nhdr;

/* The buffer is pointing to the start of the section/segment. */
   note = memory;

   /* If the name is defined, it follows the note. */
   name = note->n_namesz == 0 ? NULL : memory + sizeof(*note);

   /* If the descriptor is defined, it follows the name
      (with alignment). */

   desc = note->n_descsz == 0 ? NULL :
          memory + sizeof(*note) + ALIGN_UP(note->n_namesz, 4);

   /* The next note follows both (with alignment). */
   next_note = memory + sizeof(*note) +
                        ALIGN_UP(note->n_namesz, 4) +
                        ALIGN_UP(note->n_descsz, 4);
```

- **Inspiration: Systemtap drace markers https://github.com/jav/systemtap/blob/master/includes/sys/sdt.h**

Information in a Hypothetical Function Note

- **The basics**
  - Location of thread or initialization function in binary
    - SDT inserts a no-op in the beginning of a function and records a note that points to it
    - could use a similar strategy to relate note back to thread function
  - Encoding information in a function note string
    - Version number
    - Property list to contain important information?
      - perhaps in json information?
- **Useful information to be encoded in a function note**
  - Thread function
    - Tool thread, tool support thread …
    - Thread created while holding a lock
    - Threads that are targets for signal handlers
  - Runtime initializer
    - e.g. zeInit, cuInit
    - when should a tool be initialized
      - e.g. before the runtime, after the runtime

Using Function Notes

- **As a tool loads a module**
  - process all notes in the module to parse and record all thread function notes
- **When a thread is created**
  - tool can use thread note to decide how to handle the thread